UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Título: **Caracterización de Documentos utilizando técnicas de Minería de Textos**

Autores: Germán Aquino

Director: Laura Lanzarini

Carrera: Licenciatura en Informática

Resumen

Obtener el conjunto de términos más representativos de un documento es una tarea importante, ya que permite caracterizarlo y simplificar los procesos de búsqueda y recuperación.

En este trabajo se presenta un nuevo método que, sin importar el idioma en el que el documento esté escrito, permite extraer el conjunto de palabras clave más adecuado. Su funcionamiento se basa en una Red Neuronal que, luego de ser entrenada, es capaz de decidir para cada término del documento si se trata de una palabra clave o no. El ingreso del documento a la Red Neuronal implicó la definición de una representación numérica adecuada que permite medir la participación de un término dentro del documento.

El método propuesto fue medido sobre una base de datos de artículos científicos publicados en WICC entre los años 1999 y 2012 demostrando un desempeño superior al de KEA, uno de los algoritmos de extracción de palabras clave más citados en la bibliografía.

Palabras Claves

- *Minería de textos*
- *Extracción de palabras clave*
- *Redes Neuronales*
- *Aprendizaje automático*

Conclusiones

- *Se desarrolló un método de extracción de palabras clave independiente del idioma utilizado.*
- *Se aplicó el método propuesto a una base de datos real con resultados satisfactorios.*
- *La precisión del método propuesto es comparable a la de uno de los métodos más conocidos dentro del área.*

Trabajos Realizados

- *Desarrollo de un método de extracción de palabras clave que construye un modelo de clasificación para los términos.*
- *Evaluación de la precisión del modelo obtenido en base a la comparación con las palabras clave elegidas manualmente por los autores de los documentos.*
- *Comparación del rendimiento obtenido con el de uno de los métodos más citados en la literatura.*

Trabajos Futuros

- *Enriquecer la representación de los términos y analizar la importancia de cada atributo en la clasificación de un término como palabra clave.*
- *Aplicar algoritmos de aprendizaje automático más complejos con el fin de obtener modelos más potentes.*
- *Estudiar representaciones semánticas y gramaticales del texto que permitan mejorar la precisión de los modelos obtenidos y a realizar un análisis del sentido del texto.*

Fecha de la presentación: **DICIEMBRE 2013**

Caracterización de Documentos utilizando técnicas de Minería de Textos

AUTOR

Germán Aquino

DIRECTORA

Prof. Lic. Laura C. Lanzarini

Trabajo final presentado para obtener el grado de
Licenciado en Informática

FACULTAD DE INFORMATICA
UNIVERSIDAD NACIONAL DE LA PLATA

5 de diciembre de 2013

Objetivo

El avance de la tecnología ha posibilitado la generación de grandes repositorios de datos en distintos formatos entre los cuales el texto es el más habitual.

Esta tesina presenta un nuevo método capaz de extraer en forma automática las palabras clave de un documento. Se trata de un problema de interés ya que se aplica en diversas tareas que tienen que ver con la clasificación, organización y búsqueda de documentos.

Para medir los resultados del método propuesto se ha trabajado con una base de datos formada por artículos científicos publicados en el Workshop de Investigadores en Ciencias de la Computación entre los años 1999 y 2012.

Los resultados obtenidos han sido satisfactorios ya que el método propuesto ha evidenciado una mejor performance que otros métodos existentes en la literatura.

Prefacio

Esta tesina resume la tarea de investigación que he llevado a cabo en el III-LIDI desde febrero de 2013 en el área de Minería de Textos. Es una temática de gran auge en la actualidad, debido a la fuerte presencia de las redes sociales y a la abundancia y disponibilidad de información escrita en la Web, con lo que la necesidad de contar con herramientas que permitan aprovechar esta enorme cantidad de información es cada vez mayor.

Desde el punto de vista académico, la minería de textos presenta desafíos y problemas difíciles pero a la vez muy interesantes, como el procesamiento del lenguaje natural, en el que todavía hay mucho por hacer.

El objetivo de esta tesina de grado es profundizar en área de la minería de textos y aportar herramientas a la misma mediante del desarrollo de un método de extracción automática de palabras clave a partir de documentos de texto libre. Para ello he estudiado los distintos enfoques existentes centrándome en las aproximaciones que utilizan aprendizaje automático.

Los documentos caracterizados a través del método propuesto han sido obtenidos a través del Proyecto de Enlace de Bibliotecas de la Universidad Nacional de La Plata (PrEBi) del Servicio de Difusión de la Creación Intelectual (SeDiCI), el repositorio institucional de la Universidad Nacional de La Plata.

A lo largo del presente trabajo he descrito los aspectos centrales de las técnicas empleadas así como las transformaciones realizadas sobre los documentos. Estas últimas se relacionan con la operatoria de las técnicas específicas de extracción de conocimiento utilizadas.

Como trabajo futuro es mi intención explorar técnicas no supervisadas y representaciones alternativas de los documentos con el objetivo de capturar el significado de lo que está escrito en el texto.

Índice general

Capítulo 1

Minería de Datos y Minería de Textos

La tareas llevadas a cabo a lo largo de esta tesina forman parte de lo que habitualmente se denomina Proceso de Extracción de Conocimiento o *KDD (Knowledge Discovery in Databases)*. Este proceso tiene como objetivo principal descubrir, a partir de la información disponible, patrones, relaciones y tendencias que sean novedosos y útiles.

En este capítulo se describen los conceptos del proceso de extracción de conocimiento, sus aplicaciones y relevancia en la actualidad. También se analizan las distintas etapas que lo componen haciendo énfasis en una de ellas, la Minería de Datos, ya que es en la que se enmarca esta tesina.

En la parte final, se introducen los conceptos de la Minería de Textos y el problema de la caracterización de documentos, que es el tema central de este trabajo.

1.1. El Proceso de Extracción de Conocimiento

El aumento del volumen y la variedad de información que se encuentra en bases de datos digitales y otras fuentes ha crecido mucho en las últimas décadas. Gran parte de esta información es histórica y representa transacciones o situaciones que se han producido. Aparte de su función de "memoria de la organización", la información histórica es útil para explicar el pasado, entender el presente y predecir el futuro. La mayoría de las decisiones de las empresas, organizaciones e instituciones se basan también en información sobre experiencias pasadas. Además, ya que los datos pueden proceder de distintas fuentes y pertenecer a diferentes dominios, es clara la necesidad de analizar los mismos para la obtención de información útil para la organización.

En muchas situaciones el método tradicional de convertir los datos en *conocimiento*

consiste en el análisis e interpretación realizados de forma manual. El especialista en la materia analiza los datos y elabora un informe o hipótesis que refleja las tendencias o pautas de los mismos. Esta forma de actuar es lenta, costosa y altamente subjetiva. De hecho, el análisis manual es impracticable en dominios donde el volumen de los datos crece exponencialmente: la enorme abundancia de datos desborda la capacidad humana de comprenderlos sin la ayuda de herramientas potentes. Consecuentemente, muchas decisiones importantes se realizan no sobre la base de la gran cantidad de datos disponibles sino siguiendo la propia intuición del usuario al no disponer de las herramientas necesarias. Este es el principal aporte y objetivo del proceso de extracción de conocimiento: obtener relaciones a partir de la información disponible sin necesidad de contar con una hipótesis previa.

El proceso de extracción de conocimiento se define como el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos. Este es un proceso complejo que incluye no sólo la obtención de los modelos o patrones, sino también la evaluación y posible interpretación de los mismos.

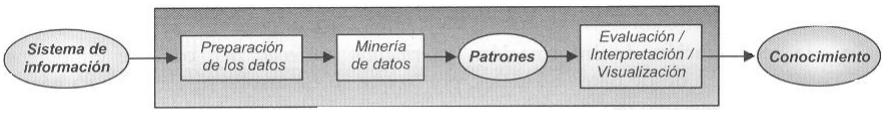

Figura 1.1: El proceso de Extracción de Conocimiento

La minería de datos es una de las etapas del proceso de KDD y engloba un conjunto de técnicas y herramientas para facilitar la extracción de conocimiento útil desde la información disponible. Se distingue de otras aproximaciones como el análisis estadístico y el procesamiento analítico *on-line* de bases de datos (*On-Line Analytical Processing, OLAP*), porque no obtiene información extensional (datos) sino intensional (conocimiento). Además este conocimiento generalmente no es una parametrización de ningún modelo preestablecido o intuido por el usuario, sino que es un modelo novedoso y original, extraído completamente por la herramienta o técnica a partir de los datos.

Como se observa en la figura 1.1, los sistemas de extracción de conocimiento permiten la selección, limpieza, transformación y proyección de los datos, analizar los datos para extraer patrones y modelos adecuados, evaluar e interpretar los patrones para convertirlos en conocimiento, consolidar el conocimiento resolviendo posibles conflictos con conocimiento previamente extraído, y hacer el conocimiento disponible para su uso. Esto clarifica la relación entre el proceso de KDD y la minería de datos. KDD es el proceso global de descubrir conocimiento útil desde las bases de datos mientras que la minería

de datos se refiere a la aplicación de diversos métodos para la obtención de patrones y modelos.

1.2. Fases del proceso de Extracción de Conocimiento

El proceso de extracción de conocimiento es un proceso iterativo e interactivo. Es iterativo porque la salida de alguna de las fases puede hacer que sea necesario volver a pasos anteriores y porque a menudo son necesarias varias iteraciones para extraer conocimiento de alta calidad. Es interactivo porque el usuario, o más generalmente un experto en el dominio del problema, debe ayudar en la preparación de los datos, validación del conocimiento extraído, etc. [23, Capítulo 2]

Este proceso se organiza en varias fases como se ilustra en la Figura 1.1. En la fase de integración y recopilación de datos se determinan las fuentes de información que pueden ser útiles y dónde conseguirlas. A continuación, se transforman todos los datos a un formato común, frecuentemente mediante un almacén de datos que consiga unificar de manera operativa toda la información recogida, detectando y resolviendo las inconsistencias. Este almacén de datos facilita enormemente la navegación y visualización previa de los datos, para discernir cuáles son los aspectos de interés que deben ser analizados.

Los datos provienen de diferentes fuentes pueden contener valores erróneos o faltantes. Estas situaciones se tratan en la fase de selección, limpieza y transformación, en la que se eliminan o corrigen los datos incorrectos y se decide la estrategia a seguir con los datos incompletos. Además, se proyectan los datos para considerar únicamente aquellas variables o atributos que son relevantes, con el objetivo de hacer más fácil la tarea propia de minería de datos y para que los resultados de la misma sean más útiles. La selección incluye tanto un filtro o fusión horizontal (filas / registros) como vertical (columnas / atributos). Las dos primeras fases se suelen englobar bajo el nombre de “preparación de datos”.

En la fase de minería de datos propiamente dicha se decide cuál es la tarea a realizar (clasificar, agrupar, etc.) y se elige el método que se va a utilizar. En la fase de evaluación e interpretación los patrones son evaluados y analizados por los expertos en el dominio del problema y de ser necesario se vuelve a las fases anteriores para una nueva iteración del proceso. Esto incluye resolver posibles conflictos con el conocimiento sobre el dominio del que se disponía anteriormente.

1.2.1. Integración y Recopilación

Las bases de datos y las aplicaciones basadas en el procesamiento tradicional de datos, que se conoce como procesamiento transaccional *on-line* (*On-Line Transaction Processing, OLTP*) son suficientes para cubrir las necesidades diarias de una organización (tales como la facturación, control de inventario, nóminas, etc.). Sin embargo, resultan insuficientes para otras funciones más complejas como el análisis, la planificación y la predicción, es decir, para tomar decisiones estratégicas a largo plazo. En estos casos y dependiendo de la aplicación, lo normal es que los datos necesarios para poder llevar a cabo un proceso de KDD pertenezcan a diferentes organizaciones, o a distintos departamentos de una misma entidad. Incluso puede ocurrir que algunos datos necesarios para el análisis nunca hayan sido recolectados en el ámbito de la organización por no ser necesarios para sus aplicaciones. En muchos casos es necesario adquirir además datos externos desde bases de datos públicas (como el censo, datos demográficos o climatológicos) o desde bases de datos privadas (como los datos de compañías de pagos, bancarias, eléctricas, etc.). Esto representa un reto, ya que cada fuente de datos usa diferentes formatos de registro, diferentes grados de agregación de los datos, diferentes claves primarias, etc. Por lo tanto, el primer paso es integrar estos datos.

La integración de múltiples bases de datos ha dado lugar a la tecnología de almacenes de datos (*data warehouses*). Este término hace referencia a la tendencia actual en las empresas e instituciones de coleccionar datos de las bases de datos transaccionales y otras fuentes diversas con el fin de hacerlos accesibles para el análisis y la toma de decisiones.

Un almacén de datos es un repositorio de información coleccionada desde varias fuentes, almacenada bajo un esquema unificado que normalmente reside en un único lugar. Existen varias formas de mezclar las distintas bases de datos para crear el repositorio. Una posibilidad es simplemente hacer una copia de las bases de datos integrantes (probablemente eliminando inconsistencias y redundancias). Esta aproximación limita las ventajas de acceder a bases de datos heterogéneas. Por ello, generalmente los almacenes de datos se construyen por medio de un proceso de integración y almacenamiento en un nuevo esquema integrado. Esencialmente, los almacenes de datos se utilizan para poder agregar y cruzar eficientemente la información de formas más sofisticadas. Por ello, los datos se modelan con una estructura de base de datos multidimensional, donde cada dimensión corresponde a un atributo o conjunto de atributos en el esquema en torno a unos “hechos” que almacenan el valor de alguna medida agregada como por ejemplo la cantidad vendida de un producto en un día concreto en una tienda. Esta visión multidimensional hace a los almacenes de datos adecuados para el procesamiento analítico *on-line* (*On-Line Analytical Processing, OLAP*). Las operaciones OLAP permiten un análisis multidimensional de los datos, que es superior al SQL para elaborar

resúmenes y detalles en varias dimensiones pudiendo utilizar conocimiento previo sobre el dominio de los datos. Esto permite la presentación de los datos a diferentes niveles de abstracción, lo que a su vez permite mostrar al usuario diferentes vistas de los datos.

Es importante destacar la diferencia entre OLAP y la minería de datos. El usuario de una herramienta OLAP la utiliza para obtener información agregada a partir de información detallada, combinándola de manera flexible. Esto permite obtener informes y vistas sofisticadas en tiempo real. Además, las herramientas OLAP pueden utilizarse para comprobar rápidamente patrones y pautas hipotéticas sugeridas por el usuario con el objetivo de verificarlas o rechazarlas. Se trata, por lo tanto, de un proceso esencialmente deductivo. Por el contrario, la minería de datos, en lugar de verificar patrones hipotéticos, usa los datos para encontrar estos patrones. Por lo tanto, es un proceso inductivo. Ambos tipos de herramientas se complementan: se puede utilizar OLAP al principio del proceso de KDD para explorar los datos (por ejemplo para centrar la atención en las variables importantes, identificar excepciones o encontrar interacciones), ya que cuanto más se comprenda sobre los datos más efectivo será el proceso de descubrir conocimiento.

1.2.2. Selección, limpieza y transformación

La calidad del conocimiento descubierto no sólo depende del algoritmo de minería utilizado, sino también de la calidad de los datos sobre los que se aplica el mismo. Por ello, después de la recopilación, el siguiente paso en el proceso de KDD es seleccionar y preparar el subconjunto de datos que se va a procesar, los cuales constituyen lo que se conoce como *vista minable*. Este paso es necesario ya que algunos datos recolectados en la etapa anterior son irrelevantes o innecesarios para la tarea de minería que se quiere realizar.

Además de la irrelevancia, existen otros problemas que afectan a la calidad de los datos. Uno de estos problemas es la presencia de valores que no se ajustan al comportamiento general de los datos, los valores anómalos o *outliers*. Estos datos anómalos pueden representar errores en los datos o pueden ser valores correctos que simplemente son muy diferentes a los demás. Algunos algoritmos de minería de datos ignoran estos datos, otros los descartan considerándolos *ruido* (inexactitudes o variaciones inesperadas en los datos) o excepciones, pero otros son muy sensibles a estos valores y el resultado obtenido se ve claramente perjudicado. Sin embargo, no siempre es conveniente eliminarlos ya que en algunas ocasiones como en la detección de compras fraudulentas efectuadas con tarjetas de crédito o en la predicción de inundaciones, los eventos raros pueden ser más interesantes que los regulares (por ejemplo, compras por un importe mucho más elevado que el de las compras efectuadas habitualmente con la tarjeta, o días en los que la cantidad

de lluvia recogida es muy superior a la media).

La presencia de datos faltantes o perdidos (*missing values*) puede ser también un problema que conduzca a resultados poco precisos. No obstante, es necesario reflexionar primero sobre el significado de los valores faltantes antes de tomar una decisión sobre cómo tratarlos ya que su presencia puede deberse a causas muy diversas, como a un mal funcionamiento del dispositivo que hizo la lectura del valor, a cambios efectuados en los procedimientos usados durante la recolección de los datos o al hecho de que los datos se recopilen desde fuentes distintas.

Los valores anómalos y los datos faltantes son sólo dos ejemplos que muestran la necesidad de la limpieza de los datos, pero además es necesario proporcionar a los métodos de minería de datos el subconjunto de datos más adecuado para resolver el problema. Para ello es necesario aplicar una selección a los datos.

La selección de atributos relevantes es uno de los pre-procesamientos más importantes, ya que es crucial que los atributos utilizados sean relevantes para la tarea de minería de datos. Idealmente, podrían usarse todas las variables (atributos) y dejar que la herramienta de minería de datos fuera probando hasta elegir las mejores. Esta forma de trabajar no funciona bien, entre otras cosas porque el tiempo requerido para construir el modelo crece con el número de variables. Aunque algunos algoritmos de minería de datos automáticamente ignoran las variables irrelevantes, en la práctica el conocimiento sobre el dominio del problema puede permitir hacer muchas de estas selecciones de forma correcta.

Como en el caso de las variables, también se puede construir el modelo usando todos los datos, pero esto significa un mayor costo en tiempo de procesamiento y recursos utilizados. Consecuentemente, una idea es usar una muestra (*sample*) de los datos. La selección de la muestra debe realizarse cuidadosamente para asegurar que es verdaderamente aleatoria.

Otra tarea de preparación de los datos es la construcción de nuevos atributos a partir de los originales con el objetivo de facilitar el proceso de minería. Esto es especialmente útil cuando los atributos originales no tienen mucho poder predictivo por sí solos o cuando los patrones dependen de variaciones lineales de las variables originales. Por ejemplo, en un problema dado en el que se analicen datos demográficos podría ser ventajoso definir un atributo “densidad de población” a partir de los atributos “cantidad de habitantes” y “superficie”.

El tipo de los datos puede también modificarse para facilitar el uso de técnicas que requieran tipos de datos específicos. Así, algunos atributos se pueden numerizar, lo que reduce el espacio y permite utilizar técnicas numéricas. El proceso inverso es discretizar

los atributos continuos, es decir, transformar valores numéricos continuos en discretos o nominales. La discretización consiste en repartir los valores de un atributo continuo en una lista de intervalos de forma que cada intervalo sea visto como un valor discreto. Estas cuestiones referentes a los tipos de datos son vistas en la Sección 1.3.

1.2.3. El proceso de Minería de Datos

La fase de minería de datos es la etapa más característica del proceso de KDD y por esta razón muchas veces se utiliza esta fase para nombrar al proceso entero. El objetivo de esta fase es producir conocimiento nuevo que pueda ser utilizado por el usuario. Esto se realiza construyendo un modelo basado en los datos recopilados para este fin. El modelo es una descripción de los patrones y relaciones entre los datos que se puede usar para hacer predicciones, para entender mejor los datos o para explicar situaciones pasadas. Esto implica tomar una serie de decisiones antes de empezar el proceso:

- Determinar qué tipo de tarea de minería es el más apropiado. Por ejemplo, una entidad bancaria podría estar interesada en caracterizar a sus clientes para enviarles publicidad referida a distintos tipos de servicios. En este caso se trata de una tarea descriptiva. En cambio, si el objetivo es predecir si un cliente abandonará la entidad, la tarea a realizar podría ser de clasificación.

- Elegir el tipo de modelo. Por ejemplo, para una tarea de clasificación se podría utilizar un árbol de decisión. Si la estructura final no resulta demasiado compleja, el recorrido de sus ramas permitirá justificar la respuesta obtenida.

- Elegir el algoritmo de minería de datos que resuelva la tarea y obtenga el tipo de modelo buscado. Esta elección es pertinente porque existen muchos métodos para construir un mismo tipo de modelo. Por ejemplo, para crear árboles de decisión para tareas de clasificación se puede usar CART o C5.0, entre otros.

Dentro de la minería de datos se pueden distinguir distintos tipos de tareas, cada una de las cuales puede considerarse como un tipo de problema a ser resuelto por un algoritmo de minería de datos. Esto significa que cada tarea tiene sus propios requisitos y que el tipo de conocimiento obtenido con una u otra puede variar bastante.

Las distintas tareas de la minería de datos pueden ser predictivas o descriptivas. En las primeras hay que predecir uno o más valores para un conjunto de instancias, las cuales son filas de la base de datos. Los datos utilizados como *ejemplos* para construir el modelo van acompañados de una salida, sea una clase, una categoría o un valor numérico. En cambio,

en las tareas descriptivas los datos constituyen un conjunto sin etiquetas ni orden alguno y el objetivo no es predecir datos nuevos sino describir los existentes. Entre las tareas predictivas están la clasificación y la regresión, mientras que el agrupamiento (*clustering*), la extracción de reglas de asociación y las correlaciones son tareas descriptivas.

- La clasificación es la tarea más utilizada. En ella, cada instancia pertenece a una clase, la cual se indica mediante el valor de un atributo especial que toma uno entre diferentes valores discretos. Entre los atributos restantes de la instancia, aquellos que sean relevantes se utilizan para predecir la clase de una instancia nueva cuya clase es desconocida. El objetivo de un algoritmo de clasificación es maximizar la tasa de precisión de la clasificación de las nuevas instancias, la cual se calcula como el cociente entre las predicciones correctas y el número total de predicciones (correctas e incorrectas). En la Figura 1.2 se muestra un problema de clasificación que consiste en separar en dos clases un conjunto de ejemplos, en este caso puntos en dos dimensiones. La línea punteada representa la frontera "real" entre las clases, y la linea llena es la frontera aproximada por un clasificador lineal, que asigna una clase a los puntos situados por debajo de la recta y otra clase a los situados encima de ella.

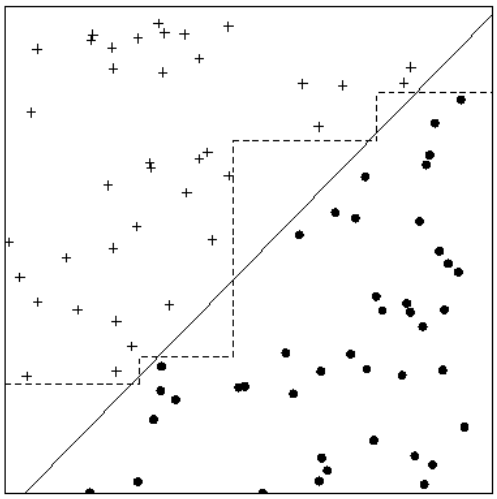

Figura 1.2: Clasificación

- La regresión (o predicción) es también una tarea predictiva que consiste en encontrar una función que asigna a cada instancia un valor real. Esta es la principal diferencia respecto a la clasificación; el valor a predecir es numérico en lugar de ser un valor discreto. El objetivo en este caso es minimizar el error (generalmente el error cuadrático medio) entre el valor predicho y el valor real.

- El agrupamiento (*clustering*) es la tarea descriptiva más común y consiste en obtener grupos "naturales" a partir de los datos. Se habla de grupos y no de clases porque, a diferencia de la clasificación, en lugar de analizar datos etiquetados con clases se los analiza justamente para generar estas etiquetas. Los datos son agrupados basándose en el principio de maximizar la similitud entre los elementos de un grupo minimizando a la vez la similitud entre grupos distintos. En otras palabras, se forman grupos tales que los elementos de un mismo grupo sean similares entre sí y al mismo tiempo sean diferentes a los elementos de otro grupo.

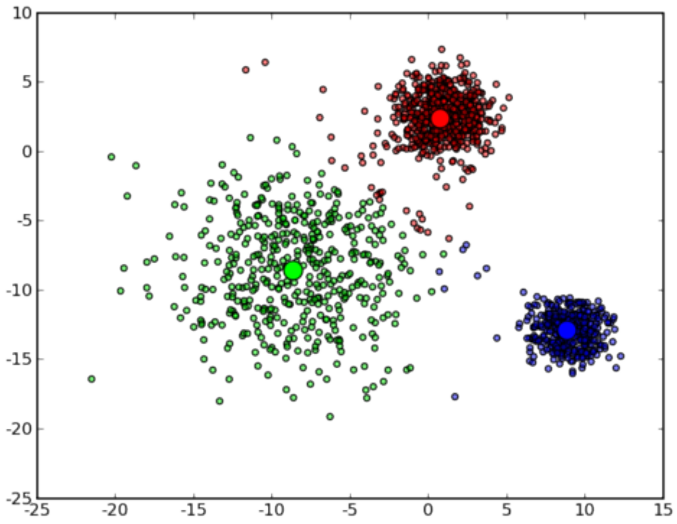

Figura 1.3: Agrupamiento (clustering)

- Las correlaciones son una tarea descriptiva que se usan para examinar el grado de similitud de los valores de dos variables numéricas. Una fórmula estándar para medir la correlación lineal es el coeficiente de correlación *r*, el cual es un valor comprendido entre -1 y 1. Si *r* es 1 o -1 las variables están perfectamente correlacionadas, ya sea positiva o negativamente, mientras que si *r* es 0 no hay correlación lineal. Este quiere decir que cuando *r* es positivo las variables tienen un comportamiento similar (ambas crecen o decrecen al mismo tiempo) y cuando *r* es negativo una variable crece cuando la otra decrece. La Figura 1.4 ejemplifica estas situaciones. El análisis de correlaciones, especialmente las negativas, puede ser muy útil para establecer reglas de items correlacionados.

- Las reglas de asociación son también una tarea descriptiva, muy similar a las correlaciones, que tienen como objetivo identificar relaciones no explícitas entre atributos nominales. Pueden ser de muchas formas, aunque la formulación más

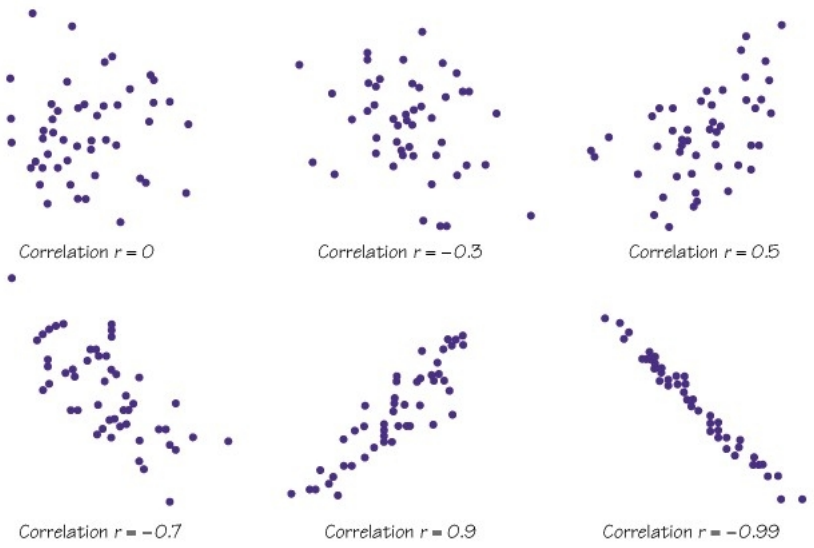

Figura 1.4: Correlaciones

común es del estilo "si el atributo X toma el valor *d* entonces el atributo Y toma el valor *b*". Las reglas de asociación no implican una relación causa-efecto, es decir, puede no existir una causa para que los datos estén asociados. Este tipo de tarea se utiliza frecuentemente en el análisis de la canasta de compra para identificar productos que son frecuentemente comprados juntos, con el fin de obtener información que pueda usarse para ajustar los inventarios, para la organización física de un almacén o en campañas publicitarias. Las reglas se evalúan usando dos criterios: precisión y soporte (o cobertura), los cuales van a ser explicados cuando se describa la fase de evaluación e interpretación del proceso de extracción de conocimiento.

Un caso especial de reglas de asociación, que recibe el nombre de "reglas de asociación secuenciales", se usa para determinar patrones secuenciales en los datos. Estos patrones se basan en secuencias temporales de acciones y difieren de las reglas de asociación en que las relaciones entre los datos se basan en el tiempo.

Técnicas de la Minería de Datos

Dado que la minería de datos es un campo interdisciplinario, existen diferentes paradigmas detrás de las técnicas utilizadas: se tienen técnicas de inferencia estadística, árboles de decisión, redes neuronales, inducción de reglas, aprendizaje basado en instancias, algoritmos genéticos, aprendizaje bayesiano, programación lógica inductiva y varios tipos de métodos basados en núcleos (*kernel*), entre otros. Cada uno de

estos paradigmas incluye diferentes algoritmos y variaciones de los mismos, así como otras restricciones que hacen que la efectividad del algoritmo dependa del dominio de aplicación, lo que resulta en que no exista un método universal aplicable a todo tipo de problema.

Existen muchos conceptos estadísticos que son la base de varias técnicas de minería de datos, como por ejemplo la **regresión lineal**. En general, la fórmula de para una regresión lineal es $y = c_0 + c_1 x_1 + \cdots + c_n x_n$, donde los x_i son los atributos predictores e y es la salida (la variable dependiente). Si los atributos son modificados en la función de regresión por alguna otra función (cuadrados, inversa, logaritmos, combinaciones de variables ...), es decir $y = c_0 + f_1(x_1) + \cdots + f_n(x_n)$, la regresión es no lineal.

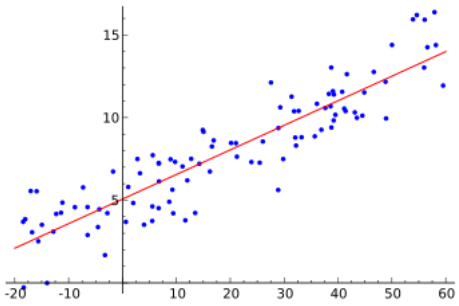

Figura 1.5: Regresión lineal

Las técnicas estadísticas no son sólo útiles para hacer regresión, sino que se utilizan también para clasificar o agrupar los datos. La inferencia de funciones discriminantes que separen clases o grupos se puede realizar de manera paramétrica o no paramétrica. Algunas de las técnicas de discriminantes no paramétricos tienen una relación muy estrecha con los métodos basados en núcleos, de los cuales las **máquinas de vectores de soporte** (*Support Vector Machine, SVM*) son el ejemplo más representativo. En estos métodos se busca un discriminante lineal que maximice la distancia entre los ejemplos fronterizos de los distintos grupos o clases. Lo interesante de estas técnicas es que en problemas no linealmente separables este discriminante lineal también se puede encontrar porque se utilizan núcleos para convertir el problema en un problema de mayor dimensionalidad. La separabilidad lineal y la dimensionalidad son tratadas en la Sección 1.3.

En otras ocasiones, dada una instancia sin clasificar se desea calcular cuál es la probabilidad de que se le asigne cada una de las clases y seleccionar aquélla que tenga la mayor probabilidad. Esta es la idea subyacente en los **métodos bayesianos**. Uno de los métodos más utilizados es el *Naive Bayes*, que se basa en la regla de Bayes y que "ingenuamente" asume la independencia de los atributos dada la clase. Es útil combinar este método con procedimientos de selección de atributos para eliminar la redundancia.

La regla de Bayes establece que, dada una hipótesis H sustentada para una evidencia E, entonces:

$$p(H|E) = \frac{p(E|H) \cdot p(H)}{p(E)}$$

donde *p(A)* representa la probabilidad del evento A, usando la notación *p(A | B)* para denotar la probabilidad del evento A condicionada al evento B.

Los **árboles de decisión** son una serie de decisiones o condiciones organizadas jerárquicamente en forma de árbol. Son muy útiles para encontrar estructuras en espacios de alta dimensionalidad y en problemas que mezclen datos categóricos y numéricos. Estas técnicas son usadas en tareas de clasificación, agrupamiento y regresión. Los árboles de decisión usados para predecir variables nominales reciben el nombre de árboles de clasificación, ya que distribuyen las instancias en clases. Cuando los árboles de decisión se usan para predecir variables continuas se denominan árboles de regresión. Los árboles de decisión siguen una aproximación "divide y vencerás" para fragmentar el espacio del problema en subconjuntos. Los nodos internos (nodos de decisión) corresponden a particiones de los datos hechas considerando atributos particulares, y los arcos que salen de un nodo corresponden a los posibles valores del atributo considerado en ese nodo. Cada arco conduce a otro nodo de decisión o a un nodo hoja. Los nodos hoja representan la clase o predicción para todas aquellas instancias que alcanzan esa hoja. De esta forma, para clasificar una instancia desconocida se recorre el árbol desde arriba hacia abajo de acuerdo a los valores de los atributos examinados en cada nodo, y cuando se llega a una hoja se asigna la clase indicada por esa hoja a la instancia. Por ejemplo, en la Figura 1.6 se muestra un árbol de decisión con el fin de asistir a un profesional de la salud en la decisión de si se debe suministrar un fármaco dado a un paciente.

Se puede considerar a los árboles como una forma de aprendizaje de reglas, ya que cada rama del árbol puede interpretarse como una regla donde los nodos internos en el camino desde la raíz a las hojas definen los términos de la conjunción que constituye el antecedente de la regla, y la clase asignada en la hoja es el consecuente.

La **inducción de reglas** es un conjunto de métodos para derivar un conjunto de reglas comprensibles de la forma:

$$SI\ (cond_1\ Y\ cond_2\ Y\ ...\ Y\ cond_n)\ ENTONCES\ pred \tag{1.1}$$

El antecedente de la regla (la parte SI) contiene una conjunción de n condiciones sobre los valores de los atributos que representan variables independientes, mientras que el consecuente de la regla (la parte ENTONCES) contiene una predicción sobre el valor

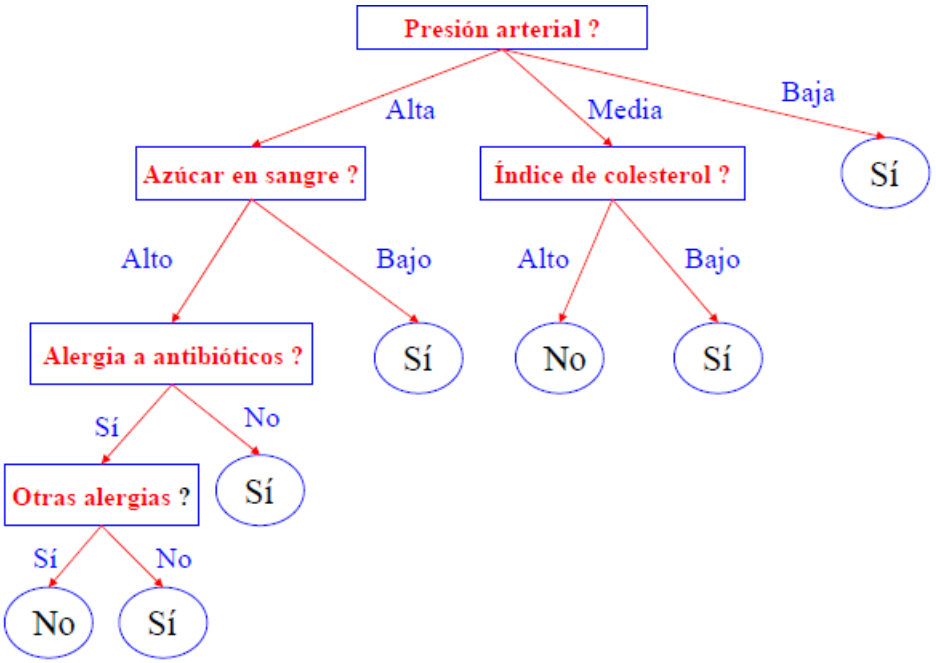

Figura 1.6: Arbol de decisión que permite determinar si debe suministrarse o no cierto fármaco a un paciente en base a los síntomas que presenta.

de una variable dependiente. La semántica de este tipo de reglas de predicción es: si se satisfacen todas las condiciones especificadas en el antecedente de la regla para los atributos independientes de un registro de datos (una instancia) entonces se predice que el atributo dependiente de este registro tendrá el valor especificado en el consecuente de la regla. Aunque los árboles de decisión también pueden producir un conjunto de reglas, los métodos de inducción de reglas son diferentes ya que:

- las reglas son independientes y no tienen por qué formar un árbol.
- las reglas generadas pueden no cubrir todas las situaciones posibles.
- las reglas pueden entrar en conflicto en sus predicciones; en este caso, es necesario elegir qué regla se debe seguir. Un método para resolver los conflictos consiste en asignar un valor de confianza a las reglas y usar la que tenga mayor confianza.

Algunos métodos de obtención de reglas, en especial para la tarea de reglas de asociación, se basan en conjuntos de items frecuentes (*frequent itemsets*) y utilizan técnicas de conteo y soporte mínimo para obtener las reglas.

Las **redes neuronales artificiales** son un paradigma de computación muy potente que permite modelizar problemas complejos en los que puede haber interacciones no lineales

entre variables. Como los árboles de decisión, las redes neuronales pueden usarse en problemas de clasificación, de regresión y de agrupamiento. Las redes neuronales trabajan con datos numéricos, por lo que para usarlas con datos nominales éstos deben numerizarse previamente. La numerización y la discretización son tratadas en la sección siguiente.

Una red neuronal puede verse como un grafo dirigido formado por muchos nodos o elementos de procesamiento interconectados mediante arcos dirigidos. Cada uno de estos elementos funciona independientemente de los demás, usando datos locales (la entrada y la salida del nodo) para llevar a cabo el procesamiento de los datos.

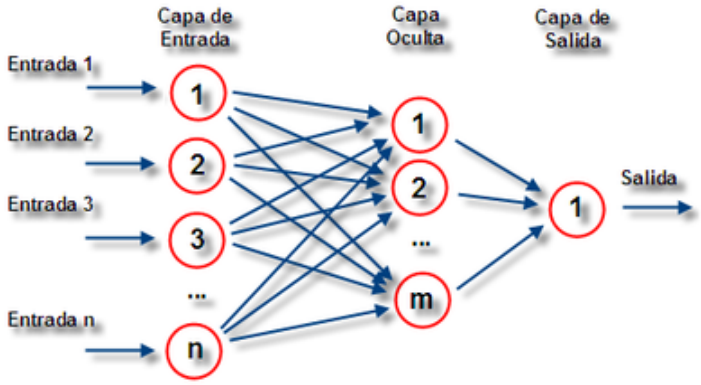

Figura 1.7: Red neuronal

La organización más popular de una red neuronal, como se observa en la Figura 1.7, consta de una capa de entrada en la que cada nodo corresponde a una variable independiente a examinar, varios nodos internos organizados en una o varias capas ocultas y una capa de salida con los nodos de salida (los posibles valores de las variables objetivo). Cada nodo de la capa de entrada está conectado a todos los nodos de la capa oculta. Los nodos de la capa oculta pueden estar conectados a los nodos de otra capa oculta o a los nodos de la capa de salida. Cada arco está etiquetado por un peso de conexión y en cada nodo hay una función de activación que indica el efecto del nodo sobre los datos que entran en él. Para aplicar una red neuronal se introducen los valores de los atributos de una instancia en los nodos de entrada, y los nodos de salida determinan la predicción para dicha instancia.

Los pesos de conexión son parámetros desconocidos que deben estimarse mediante un método de entrenamiento. El método más comúnmente utilizado es el de propagación hacia atrás (*backpropagation*), que consiste en reducir el error de la salida de la red.

Las redes neuronales tienen una gran capacidad de generalización para problemas no lineales, cuyas características van a ser vistas en la sección siguiente, pero requieren un gran volumen de datos para su entrenamiento. Su mayor desventaja es, no obstante,

que el modelo aprendido es difícil de comprender. Además, las redes neuronales sólo trabajan sobre datos numéricos. Las redes neuronales y en particular el algoritmo de *backpropagation* van a ser vistos con mayor profundidad ya que constituyen una parte importante del método presentado en esta tesina.

Los **algoritmos evolutivos** son métodos de búsqueda colectiva en el espacio de soluciones. Dada una población de potenciales soluciones a un problema, la computación evolutiva expande esta población con nuevas y mejores soluciones. El nombre se debe a que estos algoritmos siguen los patrones de la evolución biológica. Los individuos de la población se representan y codifican mediante cromosomas. Parte de los cromosomas, los genes, se usan para definir diferentes rasgos del individuo. Durante la reproducción (cruce) los genes de los padres se combinan para producir los genes de los hijos. La población va mejorando de generación en generación, porque los individuos que representan las soluciones más adecuadas al problema tienen más posibilidades de sobrevivir.

En la minería de datos, los algoritmos genéticos se pueden usar para el agrupamiento, la clasificación y para obtener reglas de asociación, así como también para la selección de atributos. En cualquiera de estos casos se comienza con un modelo o solución inicial y a través de varias iteraciones los modelos se combinan para crear modelos nuevos. Para ello se usa un función de aptitud (*fitness function*) que selecciona los mejores individuos que sobrevivirán o serán cruzados. Los distintos algoritmos genéticos difieren en la forma en que se representan los modelos, cómo se combinan los individuos en el modelo, si existen mutaciones y cómo son estas, y cómo se usa la función de adaptación. En la figura 1.8 se ven las fases de un algoritmo genético. Se comienza con una población de potenciales soluciones, a partir de la cual se seleccionan varios individuos y se los cruza. El cruce consiste en generar individuos nuevos que tomen características de sus padres. Estas características están representadas en la codificación elegida para representar los individuos. El objetivo de este procedimiento es generar una nueva población con individuos más aptos.

Los algoritmos genéticos también pueden usarse para guiar a otros algoritmos de minería de datos en el proceso de aprendizaje. Así, por ejemplo, en las redes neuronales los algoritmos genéticos pueden usarse como un medio para ajustar los pesos reemplazando al método de *backpropagation*. En este caso, los cromosomas tendrían la información de los pesos de las neuronas.

En la construcción del modelo es donde se aprecia mejor el carácter iterativo del proceso de KDD, ya que puede ser necesario explorar modelos alternativos hasta encontrar aquel que resulte más útil para resolver un problema dado. Así, una vez obtenido un modelo y partiendo de los resultados obtenidos por el mismo, podría ser deseable construir otro

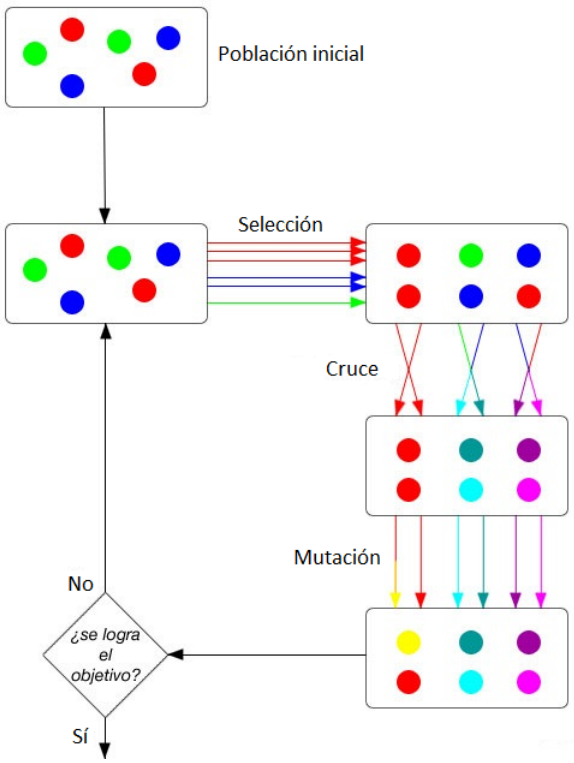

Figura 1.8: Algoritmo genético

modelo usando la misma técnica pero otros parámetros, o quizás usar otras técnicas o herramientas. En esta búsqueda del "buen modelo" puede que haya que retroceder hasta fases anteriores y hacer cambios en los datos utilizados o incluso modificar la definición del problema. Es más, la elección de la tarea a realizar y del algoritmo a emplear puede influir en la preparación de los datos (por ejemplo, un determinado algoritmo o técnica puede requerir que los datos se presenten en un formato determinado).

El proceso de construcción de modelos predictivos requiere tener bien definidas las etapas de entrenamiento y validación para asegurar que las predicciones sean robustas y precisas. La idea básica es construir el modelo con una porción de los datos (*training dataset*) y luego validarlo con el resto de los datos (*test dataset*).

1.2.4. Evaluación e interpretación

Medir la calidad de los patrones descubiertos por un algoritmo de minería de datos no es un problema trivial, ya que esta medida puede depender de varios criterios, algunos de ellos bastante subjetivos. Idealmente, los patrones descubiertos deben tener tres cualidades: ser precisos, comprensibles (es decir, inteligibles por el usuario) e interesantes

(útiles y novedosos).

Para entrenar y probar un modelo se parten los datos en dos conjuntos: uno de entrenamiento y uno de prueba. Esta separación es necesaria para garantizar que la validación de la precisión del modelo es una medida independiente. Si no se usan conjuntos diferentes de entrenamiento y prueba la precisión del modelo será sobreestimada.

En los modelos predictivos, el uso de esta separación entre entrenamiento y prueba es fácil de interpretar. Por ejemplo, para una tarea de clasificación, después de generar el modelo con el conjunto de entrenamiento éste se puede usar para predecir la clase de los datos de prueba. Entonces la tasa de *precisión* se puede calcular dividiendo el número de clasificaciones correctas por la cantidad de instancias del conjunto de prueba. La precisión es una buena estimación de cómo se comportará el modelo para datos futuros similares a los de prueba. Sin embargo esta forma de proceder no garantiza que el modelo sea correcto, sino que indica que si usamos la misma técnica con una base de datos con datos similares a los de prueba, la precisión media será bastante parecida a la obtenida con éstos. Por esta razón es importante que los ejemplos utilizados tanto para entrenamiento como para prueba sean lo bastante numerosos y representativos con el fin de que el modelo obtenido sea lo más correcto posible.

El método de evaluación básico, la validación simple, reserva un porcentaje de la base de datos como conjunto de prueba y no lo usa para construir el modelo. Este porcentaje suele variar entre el 5 % y el 50 %. La división de los datos entre los conjuntos de entrenamiento y de prueba debe ser aleatoria para que la estimación sea correcta.

Si no se dispone de una gran cantidad de datos para construir el modelo puede que no sea factible reservar parte de los mismos para la etapa de evaluación. En estos casos se usa un método conocido como validación cruzada (*cross validation*). Los datos se dividen aleatoriamente en dos conjuntos equitativos con los que se estima la precisión predictiva del modelo. Para ello, primero se construye un modelo con el primer conjunto y se usa para predecir los resultados en el segundo conjunto y calcular así una tasa de error. A continuación se construye un modelo con el segundo conjunto y se usa para predecir los resultados en el segundo conjunto y se usa para predecir los resultados del primer conjunto, obteniéndose una segunda tasa de error. Finalmente, se construye un modelo con todos los datos y se calcula el promedio de las tasas de error, que se utiliza para estimar la precisión del modelo.

El método que se usa normalmente es la validación cruzada con *n* pliegues (*n-fold cross validation*). En este método los datos se dividen aleatoriamente en *n* grupos. Un grupo se reserva para el conjunto de prueba y con los *n-1* grupos restantes se construye un modelo que es utilizado para predecir el resultado de los datos del grupo reservado. Este proceso

se repite n veces, reservando cada vez un grupo diferente para la prueba. Esto significa que se calculan n tasas de error independientes. Finalmente, se construye un modelo con todos los datos y se obtienen sus tasas de error y precisión promediando las n tasas de error disponibles. Este es el método de evaluación aplicado en esta tesina para evaluar la precisión del método propuesto.

Dependiendo de la tarea de minería de datos existen diferentes medidas de evaluación de los modelos. Por ejemplo, en el contexto de la clasificación, lo normal es evaluar la calidad del modelo obtenido con respecto a su precisión, la cual se calcula como el número de instancias del conjunto de prueba clasificadas correctamente dividido por el número de instancias totales en el conjunto de prueba. El objetivo es obtener la mayor precisión posible sobre el conjunto de prueba, ya que obtener un 100 % de precisión sobre el conjunto de entrenamiento es trivial.

En el caso de que la tarea sea de extracción de reglas de asociación, se suele evaluar de forma separada cada una de las reglas con el objeto de distinguir aquéllas que puedan aplicarse a un mayor número de instancias y que tengan una precisión relativamente alta sobre las mismas. Esto se hace en base a dos conceptos:

- *Soporte (o Cobertura)*: proporción de instancias a las que la regla se aplica y predice correctamente.
- *Confianza*: proporción de instancias que la regla predice correctamente, es decir, la cobertura dividida por el número de instancias a las que se puede aplicar la regla.

Si la tarea es regresión, es decir, la salida del modelo es un valor numérico, la manera más habitual de evaluar un modelo es mediante el error cuadrático medio del valor predicho respecto al valor que se utiliza como validación. Esto promedia los errores enfatizando las diferencias entre el valor predicho y el esperado.

Para la tarea de agrupamiento, las medidas de evaluación suelen depender del método utilizado y normalmente son función de la cohesión de cada grupo y de la separación entre grupos distintos. La cohesión y separación entre grupos se puede formalizar, por ejemplo, utilizando la *distancia* media al centro del grupo de los miembros de un grupo y la distancia media entre grupos, respectivamente. Los conceptos de distancia y de densidad son dos aspectos cruciales tanto en la construcción de modelos de agrupamiento como en su evaluación.

En los problemas de clasificación se suele usar una *matriz de confusión*, la cual muestra el recuento de casos de las clases predichas y sus valores reales. La celda (i, j) de la matriz tiene la cantidad de veces para una instancia dada el modelo predijo la clase i cuando en

realidad la misma pertenece a la clase *j*. En la diagonal de la matriz, cuando *i* y *j* son iguales, están los casos en los que el modelo acertó en la predicción. Esto permite ver de una manera más sencilla las tasas de error y acierto de cada una de las clases.

Pese a todas estas medidas, en muchos casos hay que evaluar también el contexto donde el modelo se va a utilizar. Por ejemplo, en el caso de la clasificación y las reglas de asociación, usar la precisión como medida de calidad tiene ciertas desventajas. En particular, la precisión no tiene en cuenta el problema de tener distribuciones de clases no balanceadas, es decir, tener muchas instancias de unas clases y muy pocas o ninguna de otras. El desbalance en las clases es un problema en la etapa de entrenamiento de los modelos, ya que dificulta la labor de los algoritmos de aprendizaje automático. Este es un problema que se presentó en el método propuesto en esta tesina, dado que en el conjunto de entrenamiento la cantidad de ejemplos que pertenecen a la clase de las palabras clave es mucho menor a la cantidad de ejemplos que pertenecen a la clase de los términos que no son palabras clave.

1.3. Cuestiones relevantes en la Minería de Datos

En esta sección se mencionan cuestiones a tener en cuenta a la hora de utilizar o desarrollar técnicas de minería de datos. Estas cuestiones se aplican a todas las técnicas ya que afectan el rendimiento e incluso la aplicabilidad de las mismas y también están presentes en las técnicas de minería de textos.

1.3.1. Tipos de datos. Discretización y numerización

Desde el punto de vista de las técnicas de la minería de datos se distinguen entre dos tipos de datos, los númericos y los nominales o categóricos. Los numéricos contienen valores enteros o reales, y los nominales toman valores en un conjunto finito y preestablecido de categorías. No todas las técnicas son capaces de trabajar con ambos tipos de datos. Por ejemplo los árboles de decisión ID3 sólo trabajan con atributos nominales y las redes neuronales sólo trabajan con atributos numéricos. Esto hace que pueda ser necesario aplicar un proceso de transformación y preparación de los datos, como son la discretización y la numerización. [23, Capítulo 4]

La **discretización** *(binning)* es la conversión de un valor numérico a uno ordinal (nominal ordenado), por ejemplo mediante el uso de intervalos. De ser necesario puede tratarse el atributo nuevo como si fuera de tipo nominal sin orden. La discretización puede ser aplicada también cuando:

- los errores en las mediciones pueden ser grandes o existen ciertos umbrales significativos que separan categorías.

- se quiere unificar escalas diferentes.

- la longitud de los intervalos no es uniforme y una escala discreta facilita la interpretación.

La forma más sencilla de discretizar un atributo consiste en formar intervalos de la misma longitud en el rango del mismo (pero no necesariamente con la misma cantidad de elementos) utilizando el valor máximo y el valor mínimo como referencia. Para ello se restan estos dos valores y se divide el resultado por la cantidad de intervalos deseados. Sin embargo para problemas de clasificación este tipo de discretización no tiene en cuenta la distribución de clases en los intervalos. Una discretización que en algunos casos puede resultar más adecuada es aquella en la que cada intervalo tiene un número constante de individuos o aquella que mantiene una distribución similar a la normal. En muchos de estos casos la discretización deja de ser automática, teniendo que realizarse de forma manual.

La **numerización** es el proceso inverso a la discretización, ya que consiste en convertir un valor nominal en uno numérico. Aunque es menos común que la discretización, hay casos en los que es muy útil e incluso indispensable, como por ejemplo cuando se utilizan métodos estadísticos como la regresión logística, el análisis ANOVA, los discriminantes paramétricos, etc.

En estos casos lo que se suele hacer es lo que se denomina "numerización 1 a N", que consiste en usar varios *flags* o variables indicadoras. Dado un atributo nominal a que puede tomar los valores $\{a_1, a_2, \ldots, a_n\}$ se crean n variables numéricas con valor 0 o 1 dependiendo del valor que toma el atributo a. Por ejemplo, si $a = a_2$ el atributo a_2 toma el valor 1 y todos los restantes $(a_1, a_3, ...a_n)$ toman el valor 0. Esta aproximación es útil para las técnicas de extracción de reglas, ya que los atributos nuevos representan proposiciones en sí mismos y permiten identificar más fácilmente las condiciones asociadas a los atributos más significativos.

Otra aproximación consiste en asignar un orden a los valores de un atributo nominal, lo que se denomina "numerización 1 a 1". Por ejemplo si se tienen categorías que tienen un orden natural como la edad (niño, joven, adulto, mayor) o nivel de estudios (primario, secundario, terciario, universitario) se puede asociar un valor numérico a cada una de estas categorías, como podría ser numerarlas entre 0 y 3.

1.3.2. Normalización y escalado de los datos

En algunas técnicas es necesario escalar el rango de valores que pueden tomar los atributos, especialmente en las técnicas numéricas y en las redes neuronales, ya que la diferencia entre los valores absolutos perjudica el entrenamiento o la construcción del modelo. Por ejemplo, en las técnicas basadas en distancias, como el K-medias, las diferencias en atributos de rangos muy grandes influyen más en el cálculo de la distancia entre dos ejemplos que las diferencias en atributos de rangos más chicos. El escalado también es útil para disminuir el impacto de los valores anómalos en la construcción del modelo.

La aproximación más común es la *normalización lineal uniforme*, que escala los datos entre 0 y 1 de la siguiente manera:

$$x_{norm} = \frac{x - min_x}{max_x - min_x}$$

Esto hace que los valores estén entre 0 y 1 preservando la relación entre las magnitudes originales. En algunos casos, en lugar de elegir el máximo y el mínimo natural se puede optar por agrandar o disminuir estos valores con el fin de centrar la atención en un intervalo particular o cuando se cree que la selección del máximo o del mínimo pudo haber sido afectada por la presencia de valores anómalos. En estos casos la normalización hace que algunos valores queden por debajo de 0 o por encima de 1.

Otro enfoque es aplicar un escalado *sigmoidal* o *softmax*, en el que no se aplica una transformación lineal sino una transformación más pronunciada en el centro y aplanada en los bordes utilizando una función sigmoide como la que se muestra en el capítulo siguiente, en la Figura 2.3. Esto hace que los valores de los extremos se unifiquen en 0 o en 1 y los valores centrales se distribuyan más en el intervalo.

Un tercer enfoque consiste en normalizar el rango de un atributo utilizando la media y la desviación estándar del atributo:

$$x_{norm} = \frac{x - \mu_x}{\sigma_x}$$

con lo que se obtienen valores centrados en 0. Esta aproximación tiene la ventaja de que representa mejor la relación entre los valores originales y es más tolerante a la presencia de valores anómalos. Esta normalización es utilizada en el método propuesto en esta tesina para el cálculo de uno de los atributos utilizados en la construcción del modelo.

1.3.3. Expresividad y separabilidad

La característica principal que diferencia a las distintas técnicas es la manera en que se representan los patrones aprendidos o extraídos. Es por esto que algunas técnicas son más aptas para un determinado tipo de problema que otras. Muchos de los métodos se basan en colocar fronteras entre zonas distintas, como los árboles de decisión o las redes neuronales, en los que cada zona es capaz de identificar una clase o grupo. Otros métodos no establecen fronteras explícitamente, sino que se basan en centros y clasifican o agrupan por la distancia de los ejemplos a estos centros. Todos estos métodos utilizan una noción "geométrica" del espacio n-dimensional para identificar las zonas en cuestión.

El problema es que existen patrones en los datos que no dependen de la proximidad o similitud espacial o geométrica entre los datos, y por lo tanto este tipo de patrones son más difíciles de aprender para estas técnicas. En particular, los patrones *relacionales*, es decir, aquéllos que dependen de las relaciones entre distintos atributos requieren una mayor expresividad por parte de las técnicas y los modelos generados para ser capturados. Por lo tanto se puede asociar la *separabilidad* geométrica con la facilidad de aprender patrones a partir de los datos, y se entiende la *expresividad* de una técnica o método como su capacidad de descubrir patrones complejos.

Existe un conjunto de heurísticas que permiten determinar o aproximar la complejidad geométrica de un problema, o en otras palabras, el grado de separabilidad del mismo. Si el grado de separabilidad de un problema es bajo será necesario utilizar técnicas más expresivas y menos dependientes de distancias y regiones del espacio n-dimensional. Por ejemplo, ciertos problemas de clasificación que requieren docenas de clasificadores lineales se podrían resolver mejor con clasificadores no lineales. Mientras más "relacional" y menos "geométrico" es el problema, más difícil es capturar los patrones utilizando técnicas basadas en distancias y separabilidad.

En particular, una de estas heurísticas es la *separabilidad lineal*, introducida por Minsky y Papert [34], que se define de la siguiente manera: un problema de dimensión *n* es linealmente separable si existe una función de dimensión *n-1* (una recta en espacios de dos dimensiones, un plano en espacios de tres dimensiones, y un hiperplano en espacios de más dimensiones) que divida el espacio de dimensión *n* en dos grupos o clases. Esto se puede generalizar utilizando varias funciones para separar más de dos clases.

Estas nociones son aplicables a otros problemas, además de los problemas de clasificación. Para problemas de regresión se pueden sustituir las medidas de distancia por el error cuadrático medio. Para problemas de agrupamiento se pueden usar como medidas las distancias a los centros, distancias entre grupos, etc. También se pueden aplicar a datos nominales si se definen las funciones de distancia apropiadas.

1.3.4. Dimensionalidad

Uno de los factores que más influyen en el rendimiento del aprendizaje o de la construcción del modelo es el tamaño de los datos, el cual está en función de la cantidad de ejemplos, la cantidad de atributos, y la complejidad de los datos. El número de atributos en particular es uno de los aspectos más influyentes, especialmente si hay muchos atributos que sean poco significativos para el problema a resolver, ya que se tiene un espacio de alta dimensionalidad y poca densidad. Esto dificulta el aprendizaje y perjudica a los métodos basados en distancias.

Si bien ésta es la razón principal para desear reducir la dimensionalidad, también hay otras razones como la eficiencia de la construcción del modelo o la visualización de los datos. Esto hace que en ocasiones sea necesario reducir la dimensionalidad de los datos ya sea mediante un muestreo (para reducir la cantidad de ejemplos) o mediante una selección de atributos (para reducir la cantidad de atributos). También se pueden realizar transformaciones de los datos con el objeto de reducir la dimensionalidad, combinando varios atributos en uno.

1.3.5. Medidas de distancia

Muchos de los algoritmos de la minería de datos utilizan una noción geométrica del espacio de los datos de entrada para encontrar similitudes y grupos entre los ejemplos de entrada. Los algoritmos basados en distancia, como K-medias, y las redes neuronales que realizan tareas de agrupamiento requieren de una medida de la similitud entre los ejemplos, considerados como vectores n-dimensionales. La similitud se puede formalizar mediante la distancia, siendo ésta una función que toma dos elementos del espacio de entrada y retorna un número real no negativo que representa qué tan diferentes son los elementos. Las medidas de distancia tradicionales se aplican a ejemplos cuyos atributos son numéricos.

Una medida de distancia en un conjunto S es una función $dist : S \times S \rightarrow \mathbb{R}$ que debe tener una serie de propiedades:

- $\forall x, y : dist(x, y) \geq 0$, condición de no ser negativa.
- $\forall x, y : dist(x, y) = 0$ si y sólo si $x = y$, identidad
- $\forall x, y : dist(x, y) = dist(y, x)$, propiedad de simetría
- $\forall x, y, z : dist(x, z) \leq dist(x, y) + dist(y, z)$, desigualdad del triángulo

La medida de distancia más común y más utilizada es la *distancia euclídea*, que representa la longitud de la recta que une dos puntos en un espacio euclídeo n-dimensional:

$$dist(x, w) = \sqrt{\sum_{i=1}^{n} (x_i - w_i)^2}$$

Existen muchas medidas de distancia además de la distancia euclídea e incluso para un problema particular puede ser necesario definir funciones de distancia propias. Algunas de estas medidas son:

- **Distancia de Manhattan**: se basa en la idea de recorrer el espacio sin moverse "en diagonal", y constituye la suma de las distancias absolutas para cada una de las n dimensiones.

$$dist(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- **Distancia de Chebyshev**: calcula la mayor diferencia entre todas las dimensiones.

$$dist(x, y) = max_{i=1..n}|x_i - y_i|$$

- **Distancia del coseno**: considerando a los ejemplos como vectores, es la distancia del coseno del ángulo que forman.

$$dist(x, y) = arccos\left(\frac{x \cdot y}{||x|| \cdot ||y||}\right)$$

  donde $||x||$ es el módulo del vector x, definido como

$$||x|| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- **Distancia de Mahalanobis**: esta distancia utiliza la matriz de covarianza S de los ejemplos de entrada:

$$dist(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

  La matriz de covarianza S se define a partir de la muestra $x_1, x_2, \ldots, x_n$ donde cada x_i es una instancia del conjunto de datos de entrada. Cada celda S_{ij} de la matriz se define de la siguiente manera:

$$S_{ij} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^T (x_j - \mu)$$

donde μ es el vector promedio de la muestra, que se calcula promediando para cada atributo los valores de todos ejemplos.

Es conveniente escalar los datos antes de calcular distancias ya que la disparidad en las magnitudes hace que algunos atributos influyan más que otros en el cálculo. También es recomendable tratar los valores anómalos, que pueden afectar el escalado.

El concepto de distancia también se puede aplicar sobre atributos nominales, definiendo una función apropiada. Por ejemplo, se puede definir una función de distancia como la siguiente:

$$dist(x, y) = \omega \sum_{i=1}^{n} \delta(x_i, y_i)$$

donde $\delta(a, b) = 0$ si $a = b$ y $\delta(a, b) = 1$ en caso contrario, y ω es un factor de reducción, usualmente $1/n$. De esta forma se puede utilizar una medida de distancia para los atributos numéricos, otra medida para los atributos nominales, y combinar las dos.

1.4. Minería de Textos

Dentro de la minería de datos es de especial interés la disciplina conocida como minería de textos. Esto se debe principalmente al avance tecnológico que posibilita el crecimiento de la Web y las redes sociales y la creación de grandes repositorios de textos producidos por todo tipo de usuarios. La enorme cantidad de textos disponibles genera la necesidad de diseñar técnicas y algoritmos para extraer patrones interesantes y novedosos a partir de estos datos en una forma dinámica y escalable. [23, Capítulo 21]

El objetivo de la minería de textos es el descubrimiento de nueva información a partir de colecciones de documentos de texto no estructurado, es decir texto libre, generalmente en lenguaje natural aunque también podría ser código fuente u otro tipo de información textual. Las técnicas de la minería de textos también se pueden aplicar a información semi-estructurada como son las páginas Web.

Mientras la información estructurada generalmente es manipulada a través de un sistema de base de datos, la información textual es accedida a través de un motor de búsqueda que permite al usuario encontrar la información que busca mediante una

consulta formada por palabras clave. En la minería de datos tradicional se asume que los datos están estructurados y por lo tanto la mayor parte del pre-procesamiento se enfoca en normalizar los datos, aplicar transformaciones, etc. En contraste, en la minería de textos las operaciones de pre-procesamiento se centran en la identificación y extracción de características representativas de los documentos en lenguaje natural. Estas operaciones tienen el fin de dar a la información no estructurada un formato intermedio explícito sobre el que se puedan aplicar técnicas para extraer conocimiento. De esta forma la minería de textos combina técnicas de la minería de datos, aprendizaje automático *(machine learning)*, procesamiento del lenguaje natural, recuperación de la información y representación del conocimiento, e involucra el procesamiento de colecciones de documentos (categorización de textos, extracción de información, extracción de términos), el almacenamiento del texto en representaciones intermedias y las técnicas para analizar dichas representaciones (análisis de distribución, análisis de tendencias, reglas de asociación, agrupamiento).

En la disciplina de recuperación de información *(Information Retrieval, IR)* el tema central de la investigación es la precisión (relevancia) de los documentos presentados al usuario y la eficiencia del método de búsqueda. Esta disciplina trata temas como el agrupamiento de textos, la categorización de textos, el resumen automático y los sistemas de recomendación, todas tareas de la minería de textos. Sin embargo, en la recuperación de la información el foco es el acceso a la información, mientras que la minería de textos tiene el objetivo de analizar la información y encontrar patrones. La minería de textos va más allá del simple acceso a la información ya que tiene el propósito de asistir al usuario en el proceso de analizar y asimilar esta información, y así facilitar la toma de decisiones.

La minería de textos juega un papel importante en una amplia variedad de tareas dinámicas y personalizadas de manipulación de la información, como en la ordenación en tiempo real de correo electrónico o archivos en jerarquías de carpetas, en el filtrado del correo electrónico, búsqueda estructurada y/o en los navegadores web, identificación de tópicos para soportar operaciones de procesamiento específicas a un tópico, catalogación de nuevos artículos y páginas web y en los agentes de información personal.

La minería de textos toma muchas cosas de la minería de datos, por lo cual tienen muchas similitudes. En ambas se aplican rutinas de pre-procesamiento, algoritmos de descubrimiento de patrones y herramientas de visualización para mostrar los resultados.

Sin embargo, hay una serie de características que distinguen a la minería de textos de la minería de datos tradicional. La diferencia principal es que el texto es poco denso y tiene alta dimensionalidad, lo que naturalmente afecta las técnicas de minería que se pueden utilizar sobre estos datos. Por ejemplo, un texto de cien mil palabras puede tener solamente unos pocos cientos de palabras distintas. Un cuerpo de documentos podría ser

representado como una matriz dispersa (*sparse matrix*) de tamaño $n \times d$, donde n es la cantidad de documentos y d es la cantidad de palabras distintas que aparecen en todos los documentos. La entrada (i, j) de la matriz es la frecuencia, posiblemente normalizada, de la palabra j en el documento i. El gran tamaño y la poca densidad de esta matriz inciden en el rendimiento de las técnicas de minería de datos tradicionales y generan la necesidad de considerar técnicas de reducción de dimensionalidad. La variabilidad de las frecuencias de las palabras y la longitud de los documentos también llevan a considerar varias cuestiones que involucran la representación de los documentos y la normalización de los datos, cuestiones que son críticas para las tareas de minería de textos.

1.4.1. Tareas y representaciones en la minería de textos

Algunas de las tareas principales de la minería de textos son [7, Capítulo 1]:

- Agrupamiento de texto *(Text Clustering)*: consiste en agrupar documentos, oraciones o palabras similares, usualmente identificando el tópico tratado en cada uno ellos y definiendo una medida de similitud apropiada. El agrupamiento de documentos es muy útil para organizar los textos y facilitar los procesos de búsqueda y recuperación de la información. Al igual que en la minería de datos, el agrupamiento de textos permite en cierta forma encontrar "clases" de documentos, de forma que también se puede utilizar en tareas de clasificación o como una fase previa.

- categorización *(Text Categorization)*: es la identificación de una o más categorías a las que pertenece un documento, las cuales usualmente representan los tópicos tratados en los documentos. Un enfoque al problema de la categorización es la utilización de palabras clave como categorías, de forma que describan (caractericen) brevemente el documento y permiten medir la similitud con otros documentos en base a la cantidad de palabras clave comunes.

- Clasificación *(Text Classification)*: es similar al problema de la categorización, con la diferencia que en este caso las categorías o clases son fijas y conocidas de antemano, lo que facilita la aplicación de algoritmos de aprendizaje supervisado, en el cual los ejemplos de entrenamiento van acompañados de la clase a la que pertenecen. La idea es entonces construir clasificadores que asignen palabras clave a un documento, las cuales pueden no aparecer en el mismo pero sí provienen de un vocabulario controlado.

- Extracción de información *(Information Extraction)*: como por ejemplo extracción de entidades *(Named Entity Recognition)*, sean estas personas o lugares, y sus

relaciones a partir del texto. Esto permite obtener una representación semántica que posibilita llevar a cabo un análisis más profundo del texto.

- Resumen automático del texto (*Text Summarization*): consiste en obtener una breve descripción de un documento grande o de un conjunto de documentos, preservando la semántica del contenido del texto original. Las técnicas de resumen automático se dividen en dos categorías: el resumen de extracción (*extractive summarization*) y el resumen de abstracción (*abstractive summarization*). En el resumen de extracción el objetivo es precisamente extraer unidades de información (usualmente oraciones) del texto original para construir el resumen. En cambio, en el resumen de abstracción el resumen generado puede contener información que no aparezca textualmente en el documento original. Los métodos de resumen de abstracción son más complejos, por lo que la mayoría de los métodos de resumen automático son de extracción.

- Análisis de sentimientos y opiniones *(Opinion Mining)*: consiste en interpretar el texto escrito por una persona, normalmente de redes sociales y portales de noticias, con el fin de determinar el tópico del que se está hablando y la valoración personal que el interlocutor hace sobre dicho tópico, es decir, si su opinión es positiva (favorable) o negativa (desfavorable). La obtención y procesamiento de esta información tiene muchas aplicaciones, como el *marketing* dirigido a sectores de clientes y soporte en la toma de decisiones de una organización.

En la minería de textos lo primero es representar el texto en algún formato concreto que pueda ser adecuado para los algoritmos de aprendizaje. Las representaciones más habituales son las siguientes:

- bolsa de palabras (*bag of words*): llamada también representación basada en vectores, ya que cada documento se representa como un vector de dimensión J, siendo J el número de palabras a considerar. Cada palabra constituye una componente del vector y representa una característica, la cual puede ser booleana (aparece o no aparece en el documento) o basada en frecuencias (el número de veces que ha aparecido en el documento). Es importante considerar que a través de esta representación se ignora el orden de aparición de las palabras en el texto y por ser la más simple es una de las más utilizadas.

- frases (*phrases*): esta representación consiste simplemente en considerar el documento como un conjunto de frases sintácticas, tal y como se hace en el análisis del lenguaje natural. Esta representación permite mantener el contexto en el que ocurre una palabra, hecho que se pierde en la representación anterior.

- n-gramas (*n-grams*): son secuencias de palabras de longitud n. Permiten considerar términos compuestos por más de una palabra (como "ingeniería de software"). Esta es la representación utilizada en el método propuesto en esta tesina.

- representación relacional: la representación usando lógica de primer orden permite detectar patrones más complejos. Por ejemplo, cada palabra se puede representar mediante un átomo de la forma $w_i(d, p)$, el cual es cierto cuando la palabra w_i ocurre en el documento d en la posición p.

Casi todas estas representaciones se enfrentan al problema del vocabulario, es decir, tienen errores semánticos debido a la sinonimia (palabras relacionadas con la misma temática, como "declaración" y "comunicado"), la polisemia (palabras iguales con diferente significado, como el sustantivo "frío" y el adjetivo "frío"), los lemas (palabras con el mismo radical, como "descubrir" y "descubrimiento"), entre otros.

En la mayoría de las aplicaciones es deseable dar a la información contenida en el texto una representación semántica con el fin de aplicar un proceso de análisis y minería más expresivo. Por ejemplo, representar el texto a nivel de entidades como personas, lugares y organizaciones y las relaciones entre ellos puede permitir el descubrimiento de patrones más interesantes y significativos que los obtenidos utilizando una representación más sencilla como la de *bag of words*. Sin embargo, los métodos del estado del arte del procesamiento del lenguaje natural todavía están lejos de generar una representación semántica compleja y precisa. Por lo tanto la mayoría de los métodos de minería de textos utilizan representaciones más superficiales como las basadas en palabras individuales, especialmente *bag of words* la cual, aunque resulta en la pérdida de la posición de las palabras, es generalmente mucho más simple de procesar desde el punto de vista algorítmico. En dominios específicos, como el biomédico y para ciertas tareas de minería de textos como la extracción de conocimientos de la Web, las técnicas de procesamiento del lenguaje natural tienen un rol importante a la hora de obtener una representación semántica del texto más significativa.

En ocasiones es necesario reducir el conjunto original de características que resultan de aplicar una representación dada ya que las mismas pueden llegar a ser cientos de miles, algo inabordable para muchos de los algoritmos de aprendizaje inductivos que buscan generalizar características a partir de ejemplos. El proceso de selección de características más común con el objetivo de reducir la dimensionalidad es el de eliminación de *stop-words* y aplicación de *stemming*. La eliminación de stop-words consiste en determinar las palabras comunes en un documento que no son específicas o descriptivas, como por ejemplo artículos, preposiciones o conjunciones que son palabras con poco contenido semántico por sí mismas. Por su parte, el proceso de *stemming* consiste en agrupar

diferentes palabras con la misma raíz sintáctica en una sola, con la idea de que estas palabras representan el mismo concepto. [7, Capítulo 6]

Luego de elegir una representación adecuada y aplicar las técnicas necesarias de reducción de dimensionalidad se cuenta con un conjunto de datos a los que se puede aplicar una técnica particular con el objetivo de resolver una tarea determinada, como puede ser la caracterización de documentos mediante la extracción de palabras clave.

1.4.2. Caracterización de documentos

La caracterización de documentos es el problema de extraer elementos del texto (palabras, frases, conceptos) que permitan identificarlo y describirlo, facilitando así los procesos de indexado y búsqueda. También facilita la tarea de agrupamiento si los elementos extraídos son tratados como categorías. En esta tesina se analiza particularmente la extracción de palabras clave como medio de caracterizar documentos, ya que las palabras clave permiten describir el contenido de un documento de forma breve y precisa. Así, un usuario que busca documentos pertinentes a uno o más temas o tópicos específicos puede acceder a ellos más fácilmente mediante el uso de las palabras clave adecuadas.

La caracterización de documentos es un problema de gran importancia en distintas áreas:

- Desde el punto de vista organizacional, el análisis de proyectos es un problema de interés para las organizaciones que deben seleccionar y decidir por ejemplo qué emprendimiento van a subsidiar.
- Desde el punto de vista académico puede resultar de interés analizar la producción científica de las distintas universidades nacionales con carreras similares a fin de encontrar temas de investigación comunes.
- Los repositorios bibliográficos también se verían beneficiados si contaran con herramientas de categorización automática de documentos ya que facilitaría su almacenamiento y acceso.

En los sistemas de recuperación de información *(Information Retrieval)* usualmente a cada documento de una colección se le asignan una o más palabras clave que describen su contenido. De esta forma, el sistema de IR puede recuperar documentos de acuerdo a las búsquedas del usuario, basadas en las palabras clave ingresadas por el mismo. Estas palabras clave pueden pertenecer a un vocabulario controlado, el cual es usualmente un tesauro temático jerarquizado como por ejemplo el sistema de clasificación de artículos científicos utilizado por ACM [1].

La tarea de asignar palabras clave a los documentos a partir de un vocabulario controlado recibe el nombre de indexado de textos *(text indexing)*. Si las palabras clave son vistas como categorías, el indexado de textos es considerado como una instancia del problema de categorización de textos. Esta tarea puede ser resuelta de forma automática o semiautomática, en cuyo caso el usuario encargado de asignar las palabras clave las selecciona entre un conjunto de palabras clave sugeridas por el sistema.

Existen dos grandes enfoques en el problema de la caracterización o categorización de documentos y en general en las tareas de la inteligencia artificial. El primero es el de la ingeniería del conocimiento *(knowledge engineering)*, en el cual el conocimiento de los expertos es codificado en el sistema en forma declarativa o a través de reglas de clasificación. El segundo enfoque es el del aprendizaje automático *(machine learning)* en el que se lleva a cabo un proceso inductivo que construye un clasificador a partir de un conjunto de ejemplos pre-clasificados. En el dominio particular de la minería de textos, los sistemas de ingeniería del conocimiento usualmente brindan mejores resultados que los sistemas de aprendizaje automático. Sin embargo, la desventaja principal de estos sistemas es la gran cantidad de esfuerzo manual por parte de los expertos para crear y mantener la base de conocimiento utilizada por estos sistemas. Es por esta razón que la mayor parte de la investigación reciente está enfocada en el aprendizaje automático, ya que solamente requiere un conjunto de ejemplos de entrenamiento que es más fácil de conseguir y menos costoso de producir. [14, Capítulo 4]

1.5. Resumen del capítulo

En este capítulo se presentaron las definiciones y nociones más importantes del proceso de extracción de conocimiento (KDD), la minería de datos y la minería de textos. KDD es el proceso de descubrimiento de patrones, relaciones y tendencias a partir de los datos que sean novedosos y útiles.

Se describieron las fases del proceso de KDD: integración y recopilación; selección, limpieza y transformación; minería de datos; y evaluación e interpretación. Se habló de la necesidad de contar con herramientas de minería de datos, de las tareas más habituales a realizar (clasificación, regresión, agrupamiento), y de las técnicas más utilizadas para resolver cada una de estas tareas (árboles de decisión, redes neuronales, reglas).

Es de especial interés la fase de limpieza y transformación, ya que permite dar un formato adecuado a los datos para poder aplicar una técnica específica con el fin de extraer conocimiento. En el caso particular del presente trabajo, en la fase de limpieza y transformación se busca dar a un documento una representación capaz de abstraer

las características importantes de los términos presentes en el texto con el objetivo de clasificarlos como palabras clave o no. Este proceso de transformación es descripto en detalle en el Capítulo 3.

En la fase de minería de datos propiamente dicha, el objetivo es construir un modelo de los datos que permita llevar a cabo la tarea propuesta. Es importante considerar las características de este modelo, como la comprensibilidad, la expresividad y la eficiencia, que a su vez están condicionadas por la técnica empleada y por la naturaleza de los datos sobre los cuales se aplica la misma. En este capítulo se introdujeron éstas y otras cuestiones importantes a considerar cuando se construyen o utilizan herramientas de minería de datos, como el tratamiento de tipos de datos, normalización y separabilidad lineal de los problemas. Todas estas nociones van a ser referenciadas y oportunamente expandidas en los capítulos siguientes cuando se describa el método propuesto en esta tesina.

Se destacó también la importancia de la minería de textos como disciplina, y se describieron sus conceptos básicos, representaciones para los textos, y tareas más comunes. Dentro de estas tareas se describió con mayor detalle la caracterización de documentos utilizando palabras clave. Se menciona además la importancia de contar con una representación del texto que permita capturar las características principales del problema a resolver con el fin de hallar una solución adecuada.

Capítulo 2

Técnicas de la Minería de Datos

En este capítulo se describen en mayor detalle las técnicas utilizadas en la minería de datos. Además de conocer las técnicas es necesario tener en cuenta qué tipo de problema resuelve adecuadamente cada una de ellas, por lo que también se describen en detalle las tareas más comunes llevadas a cabo por los procesos de minería de datos. El énfasis está puesto en las técnicas empleadas en la construcción del método presentado en este trabajo, las redes *backpropagation* y el algoritmo de agrupamiento K-medias.

Este capítulo está organizado de la siguiente manera. En la Sección 2.1 se describen formalmente las tareas de la minería de datos introducidas en el capítulo anterior. En la Sección 2.2 se describe la relación entre las distintas técnicas de la minería de datos y las tareas que resuelven, haciendo énfasis en las características que las diferencian, como la eficiencia, la expresividad y la comprensibilidad de los modelos obtenidos, todas mencionadas en el capítulo anterior. En la Sección 2.3 se profundiza en una de las técnicas, las redes neuronales, ya que su aplicación es fundamental en el desarrollo del método presentado en este trabajo debido a su capacidad de abstraer y generalizar características complejas de los datos. Por último, en la Sección 2.4 se describe el método K-medias, que fue utilizado en este trabajo como herramienta para reducir la dimensionalidad del problema de la extracción de palabras clave a partir de documentos.

2.1. Tareas de la Minería de Datos

Como se menciona en el Capítulo 1, las tareas de la minería de datos pueden ser predictivas o descriptivas, dependiendo del objetivo de cada una de ellas. En las primeras el objetivo es predecir características de datos nuevos que no fueron analizados en la etapa de construcción del modelo, mientras que en las últimas el objetivo es encontrar patrones,

similitudes y correspondencias en los datos de entrada.

Dentro de las tareas predictivas se encuentran:

- Clasificación: formalmente, se pueden considerar a los datos de entrada como un conjunto de pares $\delta = \{ <e,s> : e \in E,\ s \in S \}$, donde E es el conjunto de datos de entrada y S es el conjunto de valores de salida. Así los ejemplos e al ir acompañados de un valor de S, son considerados ejemplos etiquetados $<e,s>$. El objetivo es aprender una función $\lambda : E \rightarrow S$ conocida como *clasificador*, que represente la correspondencia existente en los ejemplos, es decir, que asigne para cada valor de E un único valor de S. El conjunto S es nominal, lo que quiere decir que sus elementos son una enumeración $c_1, c_2, ...c_m$ finita de elementos conocidos como clases, donde m es la cantidad de clases distintas. Cuando m es dos se trata de un problema de clasificación binaria, como es el caso del método presentado en esta tesina cuya función es determinar si un *n-grama* dado pertenece a la clase de las palabras clave o no. El clasificador debe ser capaz de determinar la clase (el valor de S) de cada nuevo ejemplo sin etiquetar.

- Clasificación suave *(soft classification)*: este problema es una especialización del problema anterior. Además de aprender una función $\lambda : E \rightarrow S$ es necesario aprender otra función $\theta : E \rightarrow \mathbb{R}$, donde $\mathbb{R}$ es el conjunto de los números reales, que represente el grado de certeza de la predicción hecha por la función λ. Esta extensión permite realizar otras aplicaciones como los *rankings* de predicciones o la selección de los n mejores ejemplos.

- Estimación de probabilidad de clasificación: es una generalización de la clasificación suave. En lugar de aprender dos funciones λ y θ se trata de aprender m funciones $\theta_i : E \rightarrow \mathbb{R}$, donde m es el número de clases. Cada función θ_i asigna a cada ejemplo e un valor real p_i que representa la probabilidad (el grado de certeza) de que e pertenezca a la clase i. Para ser consideradas probabilidades en un sentido estricto, debe cumplirse que $\forall p_i : 0 \leq p_i \leq 1$ y $\sum p_i = 1$. El conjunto de funciones θ_i se denomina estimador de probabilidad. Para predecir una clase se identifica cuál es la función θ_i que maximiza el valor p_i. En problemas donde sólo hay dos clases solamente es necesaria una de las probabilidades, ya que la otra se puede calcular como $(1 - p_i)$. En este caso no hay diferencia entre el estimador de probabilidad y el clasificador suave, ya que hay una única función θ_i.

- Categorización: A diferencia de la clasificación, en este caso la función λ puede asignar más de un valor del conjunto S a cada elemento e. Es decir, cada elemento puede pertenecer a varias categorías, por ejemplo en la categorización de documentos se puede asignar m categorías a un documento, donde cada categoría

representa una temática discutida en el mismo. Al igual que la clasificación, la categorización también tiene sus variantes suave *(soft categorization)* y de estimación de probabilidades. Es interesante destacar que un estimador de probabilidades de clasificación y un estimador de probabilidades de categorización son prácticamente idénticos, con la diferencia de que para clasificar se selecciona la clase con la mayor probabilidad y para categorizar se seleccionan las mejores k categorías o aquellas categorías cuyo grado de certeza supere cierto umbral.

- Regresión: Como en la clasificación, el objetivo es aprender una función $\lambda : E \rightarrow S$ que represente la correspondencia entre cada ejemplo de E y un único valor de salida de S. La diferencia es que en este caso S no es un conjunto nominal sino numérico y sus elementos son números enteros o reales. Ejemplos de regresión son la estimación de ventas de un producto y la predicción de unidades defectuosas de un proceso de fabricación. La regresión recibe el nombre de interpolación si se utiliza para predecir valores intermedios en una secuencia.

Dentro de las tareas descriptivas están:

- Agrupamiento *(clustering)*: el objetivo de esta tarea es obtener grupos o conjuntos entre los elementos de entrada, de forma que los elementos asignados al mismo grupo sean *similares*. La diferencia principal respecto a la tarea de clasificación es que los grupos y la pertenencia a los grupos es precisamente lo que se quiere determinar y no se sabe de antemano cómo son los grupos ni cuántos grupos hay. En algunos casos se puede proporcionar el número de grupos que se desea obtener y otras veces este número es determinado por el algoritmo de agrupamiento, según las características de los datos. La función a obtener es idéntica a la de la clasificación, $\lambda : E \rightarrow S$, con la diferencia de que los valores de S se crean durante el proceso de aprendizaje y construcción del modelo. El agrupamiento permite determinar el comportamiento de una nueva instancia a partir del grupo al que pertenece, ya que probablemente comparta características con los demás elementos del mismo. Cuando se lo utiliza dentro de un proceso de KDD, luego del agrupamiento generalmente se toman decisiones diferentes para cada grupo como por ejemplo agrupar clientes en diferentes segmentos para luego decidir la estrategia de *marketing* más adecuada para cada uno de ellos. El agrupamiento también tiene sus variantes suave *(soft)* y de estimación de probabilidad, lo que brinda mayor flexibilidad a la hora de interpretar y trabajar con los grupos formados a la vez que permite construir taxonomías o agrupamientos jerárquicos. El agrupamiento además se puede usar para otros fines, como para resumir los datos originales abstrayendo sus características principales, identificar subgrupos que tengan características específicas muy distintas del resto del conjunto de datos, y

también permite reducir la dimensionalidad de un problema y combatir el problema del desbalance de clases durante el entrenamiento de un algoritmo de aprendizaje. También el agrupamiento se puede utilizar como una primera aproximación a la clasificación si se identifica cada grupo hallado con una clase.

- Correlaciones y factorizaciones: tienen el objetivo de determinar la relevancia de los atributos, detectar atributos redundantes, detectar dependencias entre atributos y seleccionar un subconjunto de atributos a partir de su relevancia. Los métodos que resuelven estas tareas trabajan exclusivamente sobre atributos numéricos con el fin de detectar correlaciones lineales y otros tipos de relaciones, como por ejemplo relaciones de causa/efecto. Pueden utilizarse modelos de regresión (que es una tarea predictiva usada para analizar la dependencia entre dos valores) con el fin de determinar la naturaleza de estas relaciones.

- Obtención de reglas de asociación: es similar a los estudios correlaciones y factoriales con la diferencia de que pueden trabajar también con atributos nominales, que son muy frecuentes en las bases de datos. Las reglas se definen de la forma *si* $A_i = a \wedge A_j = b \wedge ... \wedge A_k = h$ *entonces* $A_r = u \wedge A_s = v \wedge ... \wedge A_z = w$ donde $A_i, A_j, ..., A_z$ son atributos y $a, b, ..., w$ son valores para estos atributos. Las reglas pueden ser dirigidas (de una sola dirección), o bidireccionales, en cuyo caso representan una doble implicación. Las condiciones que componen el antecedente de una regla no necesariamente tienen que ser siempre igualdades, también se pueden usar desigualdades. También hay reglas de asociación secuenciales, cuando la asociación no aparece en el mismo momento sino en registros sucesivos de la base de datos en un intervalo de tiempo.

- Detección de dependencias funcionales: es un caso especial de la obtención de reglas de asociación, que tiene el fin de determinar si existen atributos cuyos valores son funciones de otros atributos. Para ello es necesario tener en cuenta todos los valores posibles de los atributos analizados. Pueden ser bidireccionales, como las reglas de asociación.

- Detección de valores e instancias anómalas: si bien la detección de valores anómalos o atípicos *(outliers)* se puede realizar con el fin de limpiar y preparar los datos para poder aplicar un algoritmo de minería, también se puede utilizar para detectar comportamientos anómalos como fraudes, fallos, intrusos, accidentes y catástrofes naturales. La detección de instancias anómalas tiene el objetivo de identificar aquellas instancias que no cumplen con la medida de similitud utilizada. Para ello, en general, se tienen en cuenta un gran número de atributos. Una forma de abordar la detección de anomalías es agrupar los ejemplos y ver aquellas instancias que quedan fuera de los grupos mayoritarios. Para ello son especialmente útiles los

agrupadores suaves o los estimadores de probabilidad de agrupamiento, ya que si un ejemplo tiene baja probabilidad de agrupamiento con todos los grupos puede ser considerado un caso aislado y, por lo tanto, anómalo. También se utilizan otros métodos no necesariamente basados en la tarea de agrupamiento, como la medición de distancias. Por ejemplo, una instancia cuyo vecino más próximo esté muy lejos puede considerarse una instancia anómala.

Como se observa, varias de estas tareas están estrechamente relacionadas. Esto contribuye a que la terminología sea bastante diversa, y que a veces sea difícil especificar exactamente a qué tarea particular se refiere un término dado.

2.2. Relación entre las técnicas y las tareas

Cada una de las tareas anteriores engloba técnicas, métodos y algoritmos que las resuelven. Una tarea puede tener muchas técnicas diferentes que la resuelven y además la misma técnica o el mismo tipo de técnica puede resolver varias tareas. Esto se debe a que en el fondo la mayoría de estas tareas son aspectos diferentes del mismo concepto: el aprendizaje inductivo. El aprendizaje inductivo es un tipo especial de aprendizaje que parte de casos particulares (ejemplos) y obtiene patrones, reglas o modelos que generalizan o abstraen las características de los ejemplos estudiados. Es diferente del aprendizaje abductivo, que busca explicar causas a partir de efectos, y del aprendizaje por analogía, que va de casos particulares u otros casos particulares.

Algunas de las técnicas más comunes utilizadas para resolver las tareas descriptas anteriormente son:

- Técnicas algebraicas y estadísticas: se basan, generalmente, en expresar modelos y patrones mediante fórmulas algebraicas, funciones lineales y no lineales, distribuciones y valores estadísticos agregados como medias, varianzas, correlaciones, etc. Frecuentemente, estas técnicas obtienen patrones a partir de un modelo matemático predeterminado del cual se estiman coeficientes o parámetros a partir de los datos, por lo que se las conoce como *técnicas paramétricas*. Algunos de los algoritmos más conocidos dentro de este grupo de técnicas son la regresión lineal (global o local), la regresión logarítmica y la regresión logística. Los discriminantes lineales y no lineales, basados en funciones predefinidas, también son considerados técnicas paramétricas. No obstante, aunque el término "no paramétrico" se utiliza para englobar gran parte de técnicas provenientes del aprendizaje automático, como las redes neuronales, también existen muchas técnicas de modelización estadística no

paramétrica. Se utilizan para resolver tareas de regresión, correlación y clasificación binaria (entre dos clases). Las técnicas paramétricas son muy eficientes y en muchos casos son comprensibles. Las no paramétricas en cambio son mucho más expresivas pero menos comprensibles y son más ineficientes para grandes volúmenes de datos. Tanto las técnicas paramétricas como las no paramétricas trabajan con datos numéricos, por lo que es necesario aplicar alguna técnica de numerización para utilizarlas sobre datos nominales.

- Técnicas bayesianas: se basan en estimar la probabilidad de pertenencia a una clase o grupo mediante la estimación de las probabilidades condicionales inversas o a priori, utilizando para ello el teorema de Bayes. Algunos algoritmos muy populares son el clasificador Naive Bayes, los métodos basados en máxima verosimilitud y el algoritmo EM. Las redes bayesianas generalizan las topologías de las interacciones probabilísticas entre variables y permiten representar gráficamente dichas interacciones. Se utilizan para resolver tareas de clasificación y extracción de reglas de asociación. Las técnicas bayesianas son fáciles de usar, muy eficientes, pueden tratar cientos o miles de atributos y son muy tolerantes al ruido, sin embargo su expresividad es limitada. Requieren utilizar discretización ya que funcionan sólo sobre atributos nominales.

- Técnicas basadas en conteos de frecuencias y tablas de contingencia: estas técnicas se basan en contar la frecuencia en la que dos o más eventos se presentan conjuntamente. Cuando el conjunto de eventos posibles es muy grande, existen algoritmos que van comenzando por pares de sucesos e incrementando los conjuntos sólo en aquellos casos en que las frecuencias conjuntas superen cierto umbral. Se utilizan para resolver tareas de clasificación y extracción de reglas de asociación. Un ejemplos de este tipo de algoritmos es el algoritmo Apriori para extraer reglas de asociación.

- Técnicas basadas en árboles de decisión y sistemas de aprendizaje de reglas: son técnicas que, además de su representación en forma de reglas, se basan en dos tipos de algoritmos: los algoritmos "divide y vencerás" *(divide and conquer)* como los árboles de decisión ID3 y C4.5 (para tareas de clasificación) o el CART (para tareas de clasificación y regresión), y otros como el CN2 (clasificación e inducción de reglas). Estas técnicas son fáciles de usar, admiten atributos numéricos y nominales, tratan bien los atributos no significativos y los datos faltantes, y son tolerantes al ruido. Son bastante eficientes y obtienen buenos resultados en tareas de clasificación. Su mayor ventaja es que los modelos obtenidos son muy comprensibles por el usuario y se pueden expresar en forma de conjuntos de reglas. Una desventaja de los árboles de decisión es su limitada capacidad para capturar patrones complejos.

- Técnicas relacionales, declarativas y estructurales: la característica principal de este conjunto de técnicas es que representan los modelos mediante lenguajes declarativos, como los lenguajes lógicos, funcionales, o lógico-funcionales. Las técnicas de programación lógica inductiva *(Inductive Logic Programming, ILP)* son las más representativas y las que han dado nombre a un conjunto de técnicas denominadas colectivamente minería de datos relacional. Son técnicas muy expresivas que permiten tratar datos con estructura y capturar patrones relacionales y recursivos, así como expresar el conocimiento previo en forma de reglas. Los mayores inconvenientes de estas técnicas son la dificultad en expresar los ejemplos y la poca eficiencia.

- Técnicas basadas en redes neuronales: se trata de técnicas que aprenden un modelo mediante el entrenamiento de los pesos numéricos que conectan un conjunto de nodos o neuronas. La topología de la red y los pesos de las conexiones determinan el patrón aprendido. Existen innumerables variantes en cuanto a la organización de la topología: perceptrón, redes multicapa, redes de base radial, mapas auto-organizativos de Kohonen *(Self-Organizing Map, SOM)*, etc. con algoritmos de entrenamiento diferentes para cada topología, siendo el más conocido el algoritmo de *backpropagation*. Se utilizan para resolver tareas de clasificación, regresión y agrupamiento. Las redes neuronales requieren de cierta experiencia para obtener el mejor rendimiento ya que es necesario determinar la cantidad de neuronas ocultas, la velocidad de aprendizaje, las funciones de activación, etc. Tienen tasas de precisión muy altas y son muy expresivas ya que pueden obtener modelos no lineales. Son tolerantes al ruido y a los atributos no significativos. Sin embargo, son sensibles a los valores anómalos y necesitan muchos ejemplos para entrenar. Además el aprendizaje tiene un alto costo computacional y los modelos obtenidos no son comprensibles por un usuario, dado que la relación entre las entradas y las salidas queda codificada en los pesos de las neuronas pero esta relación no es evidente. Las redes neuronales van a ser desarrolladas en detalle en la sección siguiente por tratarse de una de las técnicas utilizadas en el trabajo presentado en esta tesina.

- Técnicas basadas en núcleos y máquinas de vectores de soporte: se trata de técnicas que intentan maximizar el margen entre los grupos o las clases formadas. Para ello se basan en unas transformaciones que pueden aumentar la dimensionalidad. Estas transformaciones son conocidas como núcleos *(kernels)*. Existen muchas variantes, dependiendo del núcleo utilizado y de la manera de trabajar con el margen. Se utilizan para resolver tareas de clasificación, regresión y agrupamiento. Son técnicas muy eficientes que permiten trabajar con datos que tengan alta dimensionalidad. Los modelos obtenidos tienen una precisión alta. El mayor inconveniente es que la

calidad de los resultados obtenidos depende de la función *kernel* utilizada. Además, esta función transforma los atributos iniciales, con lo que el modelo obtenido definido sobre estas transformaciones no es comprensible para el usuario.

- Técnicas estocásticas y difusas: son un conjunto de técnicas que junto a las redes neuronales forman lo que se denomina *soft computing*. Son técnicas no determinísticas que aproximan soluciones hasta lograr converger o hasta lograr un nivel de precisión aceptable. Los elementos aleatorios son importantes en estas técnicas, y tienen la capacidad de tratar con la incertidumbre modelándola por ejemplo con probabilidades o con lógica difusa *(fuzzy logic)*. Ejemplos de este tipo de técnicas son los algoritmos genéticos y la optimización por cúmulo de partículas *(Particle Swarm Optimización, PSO)*. Una característica de estas técnicas, especialmente de los algoritmos genéticos, es su alto costo computacional. Entre sus ventajas está la flexibilidad de cambiar la función de aptitud sin tener que cambiar el algoritmo. En cuanto a las técnicas difusas, una característica positiva es su expresividad, debido al uso de fronteras difusas en lugar de fronteras estrictas.

- Técnicas basadas en casos, en densidad o en distancia: son métodos que se basan en el concepto de *distancia* entre elementos, ya sea considerando los casos más similares, como en el método de los vecinos más próximos *(nearest neighbors)* o de una forma más sofisticada, como mediante la estimación de funciones de densidad. Además de *nearest neighbours*, algunos algoritmos conocidos son los algoritmos jerárquicos como Two-step o COBWEB, y los no jerárquicos, como K-medias. Estas técnicas son fáciles de usar en general, y son eficientes si el número de ejemplos no es excesivamente grande. Son además muy expresivas ya que se utilizan precisamente para describir los datos originales. Sin embargo, al estar basadas en distancias, estas técnicas no manejan bien los atributos no significativos ni los espacios de alta dimensionalidad. El algoritmo de K-medias en particular es explicado en mayor profundidad en la Sección 2.4 ya que se utiliza en el trabajo presentado en esta tesina para balancear las clases y reducir la dimensión del problema.

Las técnicas pueden clasificarse según si el tipo de aprendizaje utilizado es supervisado o no supervisado. Las técnicas supervisadas emplean además de los ejemplos de entrenamiento la clase a la que pertenece cada ejemplo. Entre las técnicas supervisadas están Naive Bayes, redes neuronales *backpropagation* y árboles de decisión. Las técnicas no supervisadas no utilizan información adicional de los ejemplos pero con frecuencia requieren de una medida de la *similitud* entre ellos. Entre las técnicas no supervisadas están las redes neuronales SOM, y los algoritmos K-medias y Apriori. Generalmente

las técnicas supervisadas están asociadas a las tareas predictivas y las técnicas no supervisadas están asociadas a las tareas descriptivas.

Como se observa en la tabla 2.1, la correspondencia entre tareas y técnicas es muy variada. Algunas tareas pueden ser resueltas por muchas técnicas diferentes y algunas técnicas pueden aplicarse a varias tareas. Esta diversidad es una de las razones por la que es necesario conocer las capacidades de cada técnica, los contextos donde suele funcionar mejor, la eficiencia, la robustez, etc., en definitiva, las características funcionales de cada técnica respecto a las demás. [23, Capítulo 6]

Método	Tareas Predictivas		Tareas Descriptivas		
	Clasificación	Regresión	Agrupamiento	Reglas de asociación	Correlaciones
Redes Neuronales	X	X	X		
Árboles de decisión ID3, C4.5, C5.0	X				
Árboles de decisión CART	X	X			
Regresión lineal y logarítmica		X			X
Regresión logística	X			X	
K-medias			X		
Apriori				X	
Naive Bayes	X				
Nearest neighbors	X	X	X		
Análisis factorial y de componentes principales					X
Two-step, COBWEB			X		
Algoritmos genéticos y evolutivos	X	X	X	X	X
Máquinas de vectores soporte	X	X	X		
CN2	X			X	
Análisis discriminante multivariante	X				

Tabla 2.1: Correspondencia entre las técnicas y las tareas de la minería de datos

2.3. Redes Neuronales

En esta sección se describen las características, la estructura y el funcionamiento de las redes neuronales, ya que su aplicación constituye una parte importante del método presentado en esta tesina.

Como se mencionó en el Capítulo 1, las redes neuronales son sistemas dentro del campo de la Inteligencia Artificial que pueden tener diferentes aplicaciones dependiendo de su arquitectura, es decir, de la forma en que sus componentes están organizados. Se pueden usar para el reconocimiento de patrones, compresión de información, agrupamiento, clasificación, etc. Como varias de estas tareas son también tareas de la minería de datos, las redes neuronales se pueden utilizar como herramienta dentro de la misma. [11]

Las redes neuronales tienen la finalidad de emular el cerebro humano, y parten de la presunción de que la capacidad humana de procesar la información se debe a la naturaleza

biológica del cerebro. La capacidad de procesamiento del cerebro humano tiene las siguientes propiedades que las redes neuronales intentan emular:

- Capacidad de procesar información aún en presencia de fallos.
- Posibilidad de adaptarse a entornos nuevos y aprender.
- Capacidad de trabajar con información incompleta, inconsistente o que contenga ruido.
- Un alto grado de paralelismo implícito.

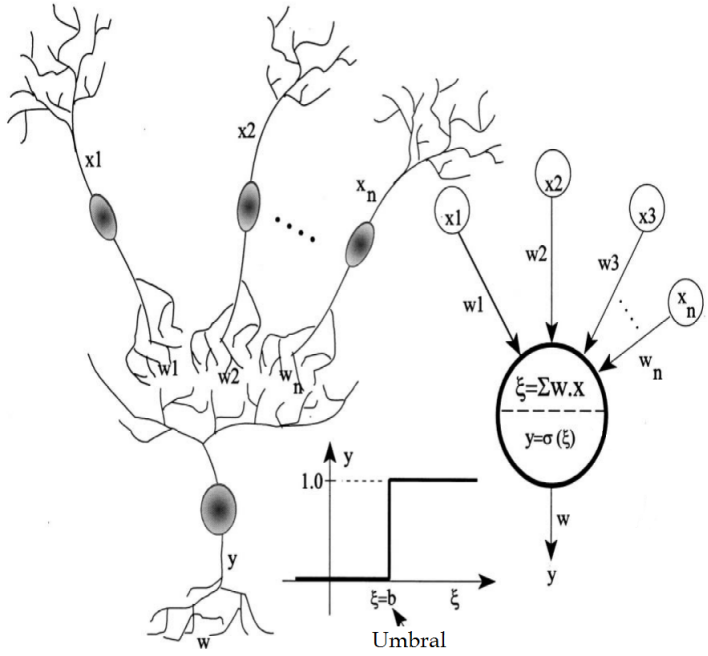

Figura 2.1: Neurona biológica y neurona artificial

Una neurona biológica recibe información a través de la sinapsis de sus dendritas. La sinapsis representa la unión de un axón de otra neurona con una dendrita, y en ella tiene lugar una transmisión electro-química que permite que la información sea transmitida desde una neurona a la próxima. La información entonces pasa de las dendritas al cuerpo de la neurona, donde los impulsos eléctricos son combinados. Si el valor resultante de esta combinación supera cierto límite o umbral, la neurona se activa enviando una señal (en forma de onda de ionización) a través de su axón con el fin de comunicarse con otras neuronas.

En base a estas características, una neurona artificial se modela como se muestra en la Figura 2.1. Las entradas son representadas mediante un vector de números reales de

entrada x, y el funcionamiento de la sinapsis (sea excitatorio o inhibitorio) se representa mediante un vector de pesos w, también de números reales. El valor de salida de la neurona viene dado por:

$$y = f\left(\sum_i w_i x_i\right) = f(\mathbf{w.x}) = f(w^T x)$$

donde f es la función de activación de la neurona.

En una red neuronal las salidas de unas neuronas se conectan con las entradas de otras. Si el peso w_i que las conecta es positivo el efecto es de excitación, en cambio si es negativo el efecto es de inhibición. El potencial de las redes neuronales radica en poder utilizar muchas unidades simples que actúen en paralelo. Normalmente las neuronas se organizan en capas como se muestra en la Figura 2.2.

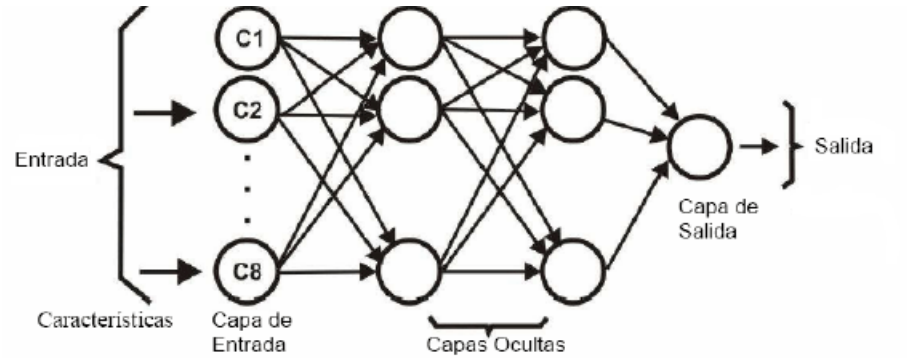

Figura 2.2: Organización en capas de una red neuronal

Así, el vector de entrada (que representa un ejemplo o instancia del conjunto de datos) ingresa a la red por la capa de entrada y se propaga por la red hasta que la activación alcanza la capa de salida. Las capas intermedias se denominan capas ocultas porque son invisibles desde afuera de la red. Las capas ocultas permiten a la red abstraer patrones más generales a partir de los datos de entrenamiento, lo que le permite aprender características más complejas.

El comportamiento de una red neuronal cambia al adaptarse al nuevo entorno, cuando se le presentan ejemplos nuevos. Estos cambios se deben a variaciones en los pesos de las neuronas como resultado de un proceso de entrenamiento, y precisamente son estos cambios los que representan el aprendizaje de la red neuronal.

Hay dos tipos de aprendizaje para redes neuronales:

- Aprendizaje supervisado: requiere que se proporcione a la red el conjunto de datos de entrada y la salida esperada para cada uno de estos datos. En un problema

de clasificación, por ejemplo, esta salida esperada es la clase a la que pertenece una instancia dada del conjunto de entrenamiento. Cada instancia del conjunto de entrenamiento se propaga por la red hasta que la activación alcanza las neuronas de la capa de salida. Entonces se compara la salida obtenida con la esperada, y se ajustan los pesos de las neuronas para asegurar que la red esté más cerca de producir una respuesta correcta en caso de que se le presente una instancia similar. Este tipo de aprendizaje se utiliza en redes que llevan a cabo tareas de clasificación y regresión. Se dará mayor énfasis a este tipo de aprendizaje ya que es el utilizado en el método propuesto en esta tesina.

- Aprendizaje no supervisado: en este tipo de aprendizaje solamente se proporciona a la red el conjunto de datos de entrenamiento, sin ninguna información adicional. La red entonces debe auto-organizarse en función de la estructura implícita de los datos de entrada, la cual está representada por similitudes existentes en los datos de entrada. Este tipo de aprendizaje se utiliza en redes que realizan tareas de agrupamiento y extracción de reglas de asociación.

El tipo de aprendizaje determina las características de las redes neuronales que resuelven estas tareas. Una propiedad importante del aprendizaje es que la red neuronal es capaz de generalizar características y dar buenos resultados en un conjunto de ejemplos de prueba que no han sido vistos durante el entrenamiento. Sin embargo, existe también el problema del sobreajuste (*overfitting*) que consiste en que la red se ajusta de una forma tan estricta a los ejemplos de entrenamiento que le es difícil reconocer correctamente instancias nuevas. El *overfitting* se caracteriza por tener una tasa de precisión muy alta para el conjunto de entrenamiento y una muy baja para el conjunto de prueba.

2.3.1. Aprendizaje supervisado

Dos de los primeros modelos de redes neuronales que emplearon el aprendizaje supervisado son el perceptrón y el combinador lineal adaptativo.

Perceptrón

El perceptrón fue inventado por Frank Rosenblatt en 1957 [38]. El perceptrón tiene una estructura de varias neuronas de entrada y una o más neuronas de salida, sin capas ocultas. La función de la capa de entrada es propagar un ejemplo de entrada x hacia la capa de salida y cada uno de ellos tiene un valor esperado de salida t. Cada neurona del perceptrón se entrena de forma independendiente, con lo que el término también se utiliza para

describir a una de estas neuronas individuales. Incluso se puede construir perceptrón formado por una única neurona.

El algoritmo de entrenamiento del perceptrón es el siguiente:

1. La red comienza en un estado aleatorio. Los pesos de las neuronas, representados con el vector w, poseen valores pequeños y aleatorios entre -1 y 1.

2. Se selecciona un vector de entrada x entre el conjunto de ejemplos de entrenamiento.

3. Se propaga la activación a través de los pesos en la red para calcular la salida $y = f(\mathbf{w.x})$. La función f se define como:

$$f(x) = \begin{cases} 1 & \text{si } w.x \geqslant \theta \\ 0 & \text{si } w.x < \theta \end{cases}$$

   En esta ecuación el término θ es el *umbral*, el valor que la entrada neta de la neurona ($w.x$) debe superar para que se produzca la activación de la neurona, lo que hace que la función f sea conocida como *función umbral*. De esta forma, la salida del perceptrón es 0 o 1, lo que lo hace adecuado para problemas de clasificación binaria.

4. Si la salida del red es correcta (es decir, si $y = t$), se vuelve al paso 2.

5. En caso contrario se actualizan los pesos de la red utilizando la siguiente fórmula: $w_{t+1} = w_t + \alpha(t-y)x$, donde w_t es el vector de pesos actual, w_{t+1} es el vector de pesos modificado, y α es un número entre 0 y 1 conocido como factor de aprendizaje. El factor de aprendizaje determina la velocidad con la que la red aprende. Un valor de α muy bajo hace que la red tarde mucho tiempo en terminar el entrenamiento, porque los pesos se modifican muy poco en cada iteración, mientras que un valor muy alto dificulta que los pesos de la red alcancen un valor óptimo, ya que los cambios de los pesos son muy grandes y por lo tanto los valores "saltan" sin poder converger.

6. Luego de modificar el vector de pesos w se vuelve al paso 2. Si ya pasaron todos los ejemplos por la red y todavía hay ejemplos mal clasificados se vuelven a ingresar los ejemplos a la red, y el algoritmo vuelve al paso 2. El entrenamiento termina cuando la salida obtenida y coincide con la salida esperada t para todos los ejemplos del conjunto de entrenamiento.

El objetivo de este algoritmo es que los pesos de la red se ajusten para clasificar correctamente los ejemplos de entrenamiento. El perceptrón es idóneo para problemas

de clasificación binaria (es decir, entre 2 clases) ya que si el problema es *linealmente separable* el perceptrón es capaz de encontrar en tiempo finito el discriminante lineal que permita clasificar correctamente todos los ejemplos. Sin embargo, si el problema no es linealmente separable el perceptrón no es capaz de llegar a una solución exacta. En estos casos es necesario detener el entrenamiento acotando la cantidad de iteraciones a realizar ya que el perceptrón no podrá proporcionar la salida correcta para todos los ejemplos de entrada.

Combinador lineal adaptativo

Otro modelo importante es el combinador lineal adaptativo (*Adaptive Linear Combiner, ALC*), desarrollado por Bernard Widrow y sus colaboradores en 1960 [42] como parte de la red neuronal Adaline *(ADAptive LINear Element)*. El Adaline es una red neuronal de una sola neurona y consiste en un ALC al que se le agrega una función umbral como la del perceptrón para realizar tareas de clasificación.

El ALC por su parte se utiliza para aproximar funciones lineales. Su topología es idéntica a la del perceptrón, con la diferencia de que para calcular la salida para un vector de entrada la función de activación f utilizada no es una función umbral sino que es la función lineal $f(x) = x$. Es decir:

$$y = \sum_{j=1}^{n} w_j x_j + b$$

donde b es conocido como el término de tendencia *(bias)*, cuya función es dar un grado de libertad a la función de aproximación para posicionarse mejor en el espacio multi-dimensional de los ejemplos de entrenamiento. El término de tendencia puede ser considerado como un peso más de la neurona, con la consideración de que no depende de un valor de entrada. Por lo tanto, sea $w = [w_1, w_2, ..., w_n]$ y $x = [x_1, x_2, ..., x_n]$, se puede considerar a b como w_0 y agregando un término $x_0 = 1$ para que los vectores tengan la misma dimensión se puede escribir la ecuación anterior como:

$$y = \sum_{j=0}^{n} w_j x_j$$

A diferencia del perceptrón, la salida del ALC es un número real, lo que le permite aproximar valores reales, como por ejemplo para realizar tareas de regresión. El Adaline utiliza el algoritmo de entrenamiento conocido como la regla del Error Cuadrático Medio *(Least Mean Square error, LMS)*, que también se conoce como la regla Delta.

El error cuadrático medio cometido por la red neuronal se define como:

$$E = \frac{\sum_{k=1}^{L} (y_k - t_k)^2}{L}$$

donde L es la cantidad de ejemplos de entrenamiento, y_k es la salida obtenida para el ejemplo k, y t_k es la salida esperada del ejemplo x_k.

El objetivo de la regla Delta es minimizar el error cuadrático medio de la red. Si se considera al error cuadrático medio como una función de n dimensiones, donde n es la cantidad de atributos de los ejemplos y por lo tanto es también la cantidad de pesos (la longitud del vector w), se puede utilizar el gradiente de esta función para encontrar el valor mínimo. El gradiente, ∇E, está formado por las n derivadas parciales de la función respecto de cada una de sus variables, que en este caso son los elementos w_i del vector de pesos w. Sin embargo, en lugar de utilizar la función de error global normalmente se utiliza la expresión de error local de cada ejemplo, es decir, $\varepsilon_k = (y_k - t_k)^2$, ya que es una función más simple de derivar y constituye una buena aproximación a la superficie de error global. Además, el uso de la expresión de error local permite que el entrenamiento de la red sea incremental, es decir, que la red observe los ejemplos uno a la vez. En caso contrario sería necesario calcular el error global para cada uno de los ejemplos de entrenamiento.

Entonces, para un ejemplo x_k dado el gradiente del error local $\nabla\varepsilon_k$ se calcula de la siguiente manera:

$$\nabla\varepsilon_k = \left[\frac{\partial(y_k - t_k)^2}{\partial w_0}, \frac{\partial(y_k - t_k)^2}{\partial w_1}, \ldots, \frac{\partial(y_k - t_k)^2}{\partial w_n}\right]$$

donde

$$\frac{\partial(y_k - t_k)^2}{\partial w_i} = 2(y_k - t_k)\frac{\partial(y_k - t_k)}{\partial w_i} = 2(y_k - t_k)\frac{\partial\left(\sum_{j=0}^{n} w_j x_{kj} - t_k\right)}{\partial w_i} \tag{2.1}$$

Como

$$\frac{\partial\left(\sum_{j=0}^{n} w_j x_{kj}\right)}{\partial w_i} = x_{ki}$$

y

$$\frac{\partial t_k}{\partial w_i} = 0$$

se tiene que

$$\frac{\partial (y_k - t_k)^2}{\partial w_i} = 2(y_k - t_k)\frac{\partial (y_k - t_k)}{\partial w_i} = 2(y_k - t_k)x_{ki}$$

con lo cual:

$$\begin{aligned} \nabla \varepsilon_k &= [2(y_k - t_k)x_{k0}, 2(y_k - t_k)x_{k1}, \ldots, 2(y_k - t_k)x_{kn}] \\ &= 2(y_k - t_k)[x_{k0}, x_{k1}, \ldots, x_{kn}] \\ &= 2(y_k - t_k)x_k \end{aligned}$$

Como se observa en la ecuación anterior, el gradiente se obtiene a partir de multiplicar el vector x_k por el error obtenido por la red, multiplicado a su vez por una constante. La idea de la regla Delta es modificar el vector de pesos w sumándole una fracción del gradiente calculado a partir de cada ejemplo de entrada con el fin de acercar a w a un valor que minimice el error cuadrático medio. Como el gradiente de una función indica la dirección de mayor crecimiento de una función, es necesario sumarle a w el gradiente negativo para acercarlo al mínimo de la función de error:

$$w_{t+1} = w_t + \alpha * (-2)(y_k - t_k)x_k$$

que también puede expresarse

$$w_{t+1} = w_t + \alpha * 2(t_k - y_k)x_k$$

La ecuación anterior es muy similar a la regla de aprendizaje del perceptrón. Al igual que en el perceptrón, el factor α representa la velocidad de aprendizaje. El algoritmo de entrenamiento ingresa los ejemplos a la red uno por vez, calculando el error para cada uno de ellos. El entrenamiento termina cuando la aproximación obtenida por la red coincide con el valor esperado para todos los ejemplos, o cuando el error acumulado de los ejemplos está por debajo de un umbral establecido.

La salida de la red es un valor real, a diferencia de la salida del perceptrón que siempre es binaria. La regla Delta también se puede aplicar para otras funciones de activación, no sólo $f(x) = x$, con la condición de que la función sea diferenciable. Con ello se obtiene un *combinador no lineal*. Por ejemplo, si se quiere utilizar un ALC para clasificar ejemplos en dos clases, se puede utilizar una función sigmoide *(logsig)* como la de la Figura 2.3:

$$logsig(x) = \frac{1}{1 + e^{-x}}$$

que es el equivalente diferenciable de la función umbral del perceptrón.

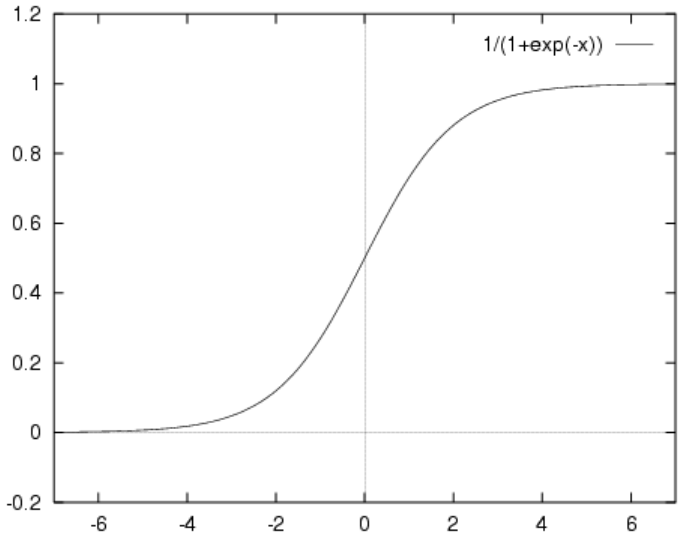

Figura 2.3: Función sigmoide

Otra función comúnmente utilizada es la tangente hiperbólica *(tansig)*, ilustrada en la Figura 2.4, cuyo rango de salida está entre -1 y 1. Esto hace que la activación de la neurona no sea de 0 y 1 sino de -1 y 1, lo que puede ser útil para tareas de clasificación en las que haya mayor separación entre las clases. Esta función se define como:

$$tansig(x) = \frac{2}{1 + e^{-2x}} - 1$$

Cuando se utiliza una función arbitraria como función de activación de la neurona, la ecuación 2.1 se puede generalizar de la siguiente forma:

$$\frac{\partial (y_k - t_k)^2}{\partial w_i} = 2(y_k - t_k)\frac{\partial (y_k - t_k)}{\partial w_i} = 2(y_k - t_k)\frac{\partial \left(f\left(\mathbf{w.x}\right) - t_k\right)}{\partial w_i} = 2(y_k - t_k) f'\left(\mathbf{w.x}\right) x_{ki} \quad (2.2)$$

donde f' es la derivada de la función de activación f respecto de w_i. Por ello es necesario que la función de activación f utilizada sea diferenciable para aplicar la regla Delta. Tanto la función sigmoide como la tangente hiperbólica son diferenciables, y además sus derivadas se pueden expresar en términos de ellas:

$$\frac{\partial logsig(x)}{\partial x} = logsig(x) * (1 - logsig(x))$$

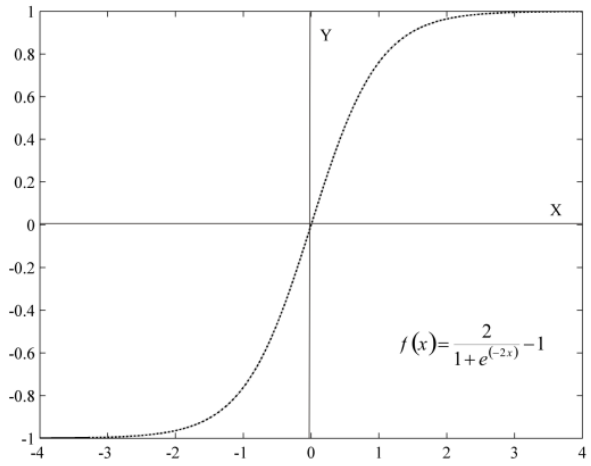

Figura 2.4: Tangente hiperbólica

$$\frac{\partial tansig(x)}{\partial x} = 1 - tansig(x)^2$$

La principal característica del ALC es el uso de una expresión de error que se intenta minimizar, lo que permite que varios ALC se integren para formar estructuras más complejas. Esto los hace adecuados para un mayor número de aplicaciones, por lo que se los utiliza para construir lo que se denomina perceptrón multicapa *(Multilayer Perceptron, MLP)*.

2.3.2. Perceptrón multicapa y algoritmo de Backpropagation

El perceptrón multicapa fue popularizado en los 80' por el Grupo PDP *(Parallel Distributed Processing)* [32] con el nombre de *backpropagation*, debido al nombre del algoritmo de aprendizaje utilizado.

En las redes multicapa la activación se propaga desde la entrada hacia la capa oculta donde se le aplica una función de activación para luego propagar la activación a la capa de salida. Esto influye en el aprendizaje de la red, ya que ahora es necesario modificar los pesos que conectan la capa de entrada con las ocultas, y los pesos que conectan las capas ocultas entre sí y con la capa de salida. El error de la capa de salida se calcula aplicando la regla Delta, y este error se "propaga hacia atrás" hacia las capas ocultas hasta llegar a la capa de entrada. Si bien este método se puede aplicar en redes con cualquier número de capas ocultas por simplicidad se va a explicar considerando una sola capa oculta.

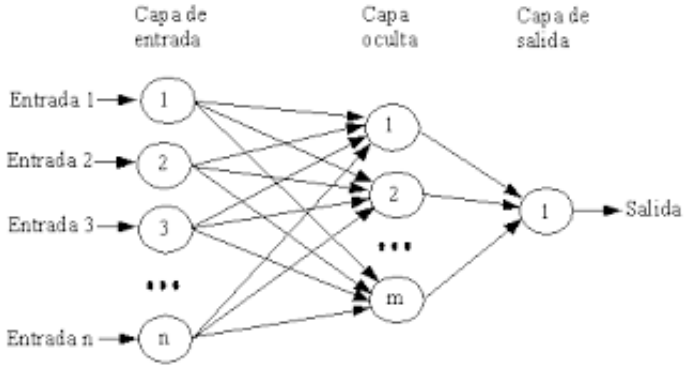

Figura 2.5: Red Backpropagation

Como se observa en la Figura 2.5 la entrada de la capa oculta es el ejemplo de entrada, y la salida de la capa oculta es la entrada de la capa de salida. Todas las neuronas de la capa de entrada se conectan con todas las neuronas de la capa oculta, y lo mismo ocurre entre las capas oculta y de salida. Por lo tanto, el gradiente del error de la capa oculta se puede calcular como en la ecuación 2.2, con la diferencia de que se utiliza la salida de la capa oculta en lugar del ejemplo de entrada.

Entonces, considerando una red *backpropagation* con m neuronas en la capa oculta, se puede definir la salida de j-ésima neurona de la capa oculta para un ejemplo x como:

$$neta_j^h = \sum_{i=0}^{n} w_{ji}^h x_i$$

$$y_j^h = f_j^h\left(neta_j^h\right) \tag{2.3}$$

y la salida de k-ésima neurona de la capa de salida a partir de la salida de la capa oculta como:

$$neta_k^o = \sum_{j=0}^{m} w_{kj}^o y_j^h$$

$$y_k^o = f_k^o\left(neta_k^o\right) \tag{2.4}$$

donde:

- w_{ji}^h es el valor del i-ésimo peso de la j-ésima neurona de la capa oculta h,

- $neta_j^h$ es la entrada neta de la j-ésima neurona de la capa oculta h,
- f_j^h es la función de activación de la j-ésima neurona de la capa oculta h, y $f_j^{h\prime}$ es la derivada de dicha función,
- w_{kj}^o es el valor del j-ésimo peso de la k-ésima neurona (que se conecta con la salida de la j-ésima neurona de la capa oculta anterior) de la capa de salida o,
- $neta_k^o$ es la entrada neta de la k-ésima neurona de la capa oculta o, y
- f_k^o es la función de activación de la k-ésima neurona de la capa oculta o, y $f_k^{o\prime}$ es la derivada de dicha función.

Usando estas definiciones y la ecuación 2.2 se calcula cada posición j del gradiente del error de la capa de salida de la siguiente forma:

$$\frac{\partial(y_k^o - t_{xk})^2}{\partial w_{kj}} = 2(y_k^o - t_{xk})\frac{\partial(y_k^o - t_{xk})}{\partial w_{kj}} = 2(y_k^o - t_{xk}) f_k^{o\prime}\left(neta_k^o\right) y_j^h$$

y por lo tanto el gradiente del error de la neurona k de la capa de salida es

$$\nabla\, \varepsilon_k^o = 2(y_k^o - t_{xk}) f_k^{o\prime}\left(neta_k^o\right) y^h \tag{2.5}$$

donde t_{xk} es la posición k del vector esperado t correspondiente al ejemplo de entrada x, e y^h representa al vector formado por las salidas de todas las neuronas de la capa oculta.

Para calcular el error de las neuronas de la capa oculta se utiliza el error de todas las neuronas de la capa de salida:

$$\varepsilon = \sum_k \left(y_k^o - t_{xk}\right)^2$$

El objetivo es hallar el valor del vector de pesos w_j^h de la j-ésima neurona de la capa oculta que minimice esta expresión de error. Por lo tanto, se deriva esta expresión respecto de cada elemento del vector w_j^h para hallar el gradiente:

$$\begin{aligned}
\frac{\partial\left(\sum_k \left(y_k^o - t_{xk}\right)^2\right)}{\partial w_{ji}^h} &= \sum_k \left[2\left(y_k^o - t_{xk}\right)\frac{\partial y_k^o}{\partial w_{ji}^h}\right] \\
&= \sum_k \left[2\left(y_k^o - t_{xk}\right) f_k^{o\prime}\left(neta_k^o\right)\frac{\partial neta_k^o}{\partial w_{ji}^h}\right] \\
&= \sum_k \left[2\left(y_k^o - t_{xk}\right) f_k^{o\prime}\left(neta_k^o\right) w_{kj}^o \frac{\partial y_j^h}{\partial w_{ji}^h}\right] \\
&= \sum_k \left[2\left(y_k^o - t_{xk}\right) f_k^{o\prime}\left(neta_k^o\right) w_{kj}^o f_j^{h\prime}\left(neta_j^h\right)\frac{\partial neta_j^h}{\partial w_{ji}^h}\right] \\
&= \sum_k \left[2\left(y_k^o - t_{xk}\right) f_k^{o\prime}\left(neta_k^o\right) w_{kj}^o f_j^{h\prime}\left(neta_j^h\right) x_i\right] \\
&= 2x_i f_j^{h\prime}\left(neta_j^h\right)\sum_k \left[\left(y_k^o - t_{xk}\right) f_k^{o\prime}\left(neta_k^o\right) w_{kj}^o\right]
\end{aligned}$$

Con esto se tiene que el gradiente del error de la neurona j de la capa oculta es:

$$\nabla\, \varepsilon_j^h = 2x f_j^{h\prime}\left(neta_j^h\right)\sum_k \left[\left(y_k^o - t_{xk}\right) f_k^{o\prime}\left(neta_k^o\right) w_{kj}^o\right] \tag{2.6}$$

Utilizando las expresiones obtenidas en 2.5 y 2.6, el algoritmo de *backpropagation* actualiza los pesos de la red de la siguiente forma:

$$\begin{aligned}
w_{k(t+1)}^o &= w_{k(t)}^o + \alpha * (-\nabla\, \varepsilon_k^o) \\
w_{j(t+1)}^h &= w_{j(t)}^h + \alpha * (-\nabla\, \varepsilon_j^h)
\end{aligned}$$

De esta forma, al descender por la superficie del error, los pesos de la red se ajustan para producir la salida esperada a partir de los ejemplos de entrada. La red entrenada puede entonces ser tratada como una función que convierte elementos de entrada x en elementos de salida y. Luego, dependiendo de la naturaleza de los x y de los y esta función puede representar por ejemplo una relación de instancia-clase, o incluso puede ser utilizada para aproximar funciones difíciles de calcular pero que tienen salidas conocidas para entradas determinadas. Las redes *backpropagation* son tolerantes al ruido y son buenas para generalizar características. Esto significa que dados varios ejemplos de entrada distintos la red aprende a encontrar las similitudes entre ellos y a reconocer correctamente ejemplos no vistos durante el entrenamiento que tengan estas características. A través del entrenamiento iterativo la red aprende a distinguir los atributos relevantes de los ejemplos

y a ignorar los no relevantes para el problema. Sin embargo, si la red es entrenada con una cantidad reducida de ejemplos o estos son poco representativos para el problema los resultados obtenidos no serán buenos. Por ejemplo, en un problema de clasificación, es importante que la red entrene con cantidades similares de datos pertenecientes a cada una de las clases.

El número de neuronas ocultas tiene un importante efecto en la capacidad de generalización de la red, ya que las redes grandes tienen mayor capacidad de "memorizar" los datos de entrenamiento. La generalización no sólo es importante para que la red funcione adecuadamente ante un conjunto de datos nuevo distinto al de entrenamiento, sino también porque es posible que haya datos que contengan ruido, que estén distorsionados o que estén incompletos. Sin embargo es necesario que exista un balance entre la precisión obtenida sobre el conjunto de entrenamiento y la obtenida sobre el conjunto de prueba. Si una red tiene demasiadas neuronas existe el riesgo de sobreajustar la red a los ejemplos de entrenamiento, lo que provocaría malos resultados sobre el conjunto de prueba. Generalmente la capa oculta tiene menos nodos que la capa de salida pero el mejor rendimiento posible depende de la naturaleza del problema en cuestión, con lo que encontrar la cantidad justa de neuronas ocultas depende en muchos casos de la experiencia y de realizar varias pruebas [16, Capítulo 3].

El tipo de tarea que la red va a realizar determina las funciones de activación que van a utilizarse y con ello la representación de los ejemplos de entrada. Por ejemplo, como se menciona en la Sección 2.3.1, para resolver un problema de clasificación se puede utilizar una función sigmoide como la de la Figura 2.3, cuyo rango de salida está acotado entre 0 y 1, mientras que para un problema de regresión en el que se necesita aproximar un valor real se puede utilizar la función lineal $f(x) = x$ cuyo rango de salida no está acotado. En cuanto a la representación, puede ser necesario aplicar una transformación a los datos de entrada con el fin de normalizarlos o escalarlos preservando la relación entre los valores de cada atributo. Por ejemplo, para aplicar la función sigmoide es necesario que la entrada esté entre 0 y 1 ya que los valores fuera de este rango producen salidas constantes que no representan la relación entre las entradas originales. En estos casos se puede aplicar alguna técnica de escalado de los datos como las descritas en el capítulo anterior.

2.3.3. Aprendizaje no supervisado

El aprendizaje no supervisado se caracteriza por descubrir modelos o características significativas a partir de los datos de entrada sin utilizar información adicional. Por lo tanto no existe, como en el aprendizaje supervisado, un valor de salida o una clase para cada instancia contra la cual contrastar la diferencia y obtener una expresión de

error. En estos casos la red neuronal debe auto-organizarse y para ello debe actuar ante algunos aspectos del conjunto de entrada como pueden ser la existencia de redundancia o similitudes en el conjunto de datos.

Dentro del aprendizaje no supervisado se utilizan dos métodos: el aprendizaje de Hebb [21], que se usa para obtener proyecciones o compresiones óptimas de conjuntos de datos, de modo similar al análisis de componentes principales; y el aprendizaje competitivo, que se utiliza principalmente para agrupar conjuntos de datos. En el aprendizaje de Hebb la idea principal es reforzar las conexiones que se utilizan frecuentemente, al igual que ocurre en las neuronas biológicas, de forma que la activación simultánea de dos neuronas conectadas aumenta el valor de los pesos que las conectan. Un ejemplo representativo de aprendizaje de Hebb son las *outstars* de Stephen Grossberg [18]. En cambio, en el aprendizaje competitivo las neuronas compiten para decidir cuál de ellas es la que se activa. Dos ejemplo de redes que utilizan el aprendizaje competitivo son las redes de Kohonen, también conocidas como mapas auto-organizativos *(Self-Organizing Map, SOM)* y las redes de contra-propagación *(Counter-Propagation Network, CPN)*, si bien estas últimas tienen un componente de aprendizaje de Hebb.

Los **mapas auto-organizativos** fueron desarrollados por Teuvo Kohonen en 1982 [27]. La idea detrás de estas redes es hallar una representación espacial de las características de los datos de entrada mediante la distribución de las neuronas en un espacio n-dimensional utilizando el concepto de *distancia*. Los mapas auto-organizativos se componen de dos capas, la capa de entrada y la capa *competitiva*. Las neuronas de la capa competitiva se organizan usualmente en topologías lineales o bidimensionales en forma de cuadrícula. El mecanismo de aprendizaje en estas redes consiste en encontrar una neurona ganadora j y actualizar sus pesos para hacer que ésta tenga más probabilidad de ganar cuando una entrada similar se presente en la red, de la siguiente manera:

$$w_{ji(t+1)} = w_{ji(t)} + \alpha(x_i - w_{ji(t)}) \tag{2.7}$$

La neurona ganadora es aquella que está más cerca del ejemplo de entrada para una medida de *distancia* dada. El factor de aprendizaje α se utiliza de la misma forma que en las redes comentadas en la sección anterior. Como la actualización de los pesos es una función de la diferencia entre la entrada y los pesos, esta regla desplaza los pesos hacia la entrada. Por lo tanto es necesaria una medida de distancia en este espacio como puede ser la distancia euclídea.

Cuando una neurona gana, además de actualizar sus pesos se actualizan también los pesos de sus vecinas en la topología de la red mediante una función de vecindad *(neighbourhood function)*. Esta función de vecindad está representada por la cantidad de neuronas que hay

en la topología entre una neurona dada y la neurona ganadora. A medida que avanza el entrenamiento la vecindad se reduce hasta que abarca una única neurona. Así, al comienzo cada neurona vecina de la ganadora se desplaza ligeramente hacia el ejemplo de entrada, hasta que el entrenamiento alcanza un punto en el que cada vector de pesos de cada neurona converge a un valor representativo de las coordenadas n-dimensionales de los ejemplos que se encuentran cerca de ella. También se puede reducir gradualmente el factor de aprendizaje, permitiendo que las neuronas se desplacen rápidamente en etapas tempranas del entrenamiento para cubrir el espacio de entrada y que se acomoden mejor a cada grupo en las etapas finales cuando los desplazamientos de los vectores de pesos son más reducidos. [25, Capítulo 6]

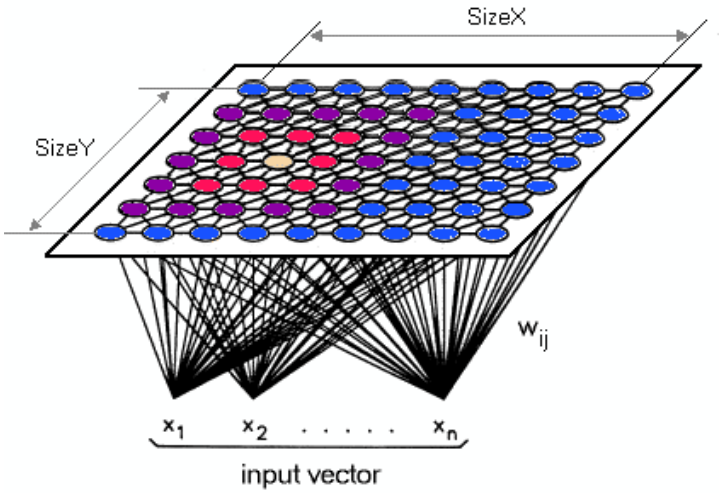

Figura 2.6: Mapa auto-organizativo (SOM)

La red obtenida al final del entrenamiento puede ser utilizada para realizar tareas de agrupamiento, identificando cada neurona con un grupo o *cluster* de elementos, y también puede ser utilizada para tareas de clasificación si se obtiene la neurona ganadora para un nuevo ejemplo y se designa como clase al grupo representado por la neurona.

Las redes de **contra-propagación** por su parte tienen el propósito de asociar pares de vectores $(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L)$ de forma de encontrar una función que dado x produzca y y posiblemente la función inversa que dado y produzca x. Estas redes fueron concebidas por Robert Hecht-Nielsen en 1987 [22] combinando las redes de Kohonen con la estructura *outstar* de Grossberg.

Una red de contra-propagación está formada por tres capas: la capa de entrada, una capa oculta y la capa de salida. Las capas oculta y de salida utilizan algoritmos de entrenamiento diferentes, por lo que el entrenamiento de la red de contra-propagación se realiza en dos fases independientes. En la primera fase se entrena la capa oculta formada

por neuronas competitivas y en la segunda fase se entrenan los pesos que conectan esta capa oculta con la de salida.

Como el objetivo de la red de contra-propagación es asociar pares de vectores (x, y), se puede considerar a y como la salida esperada de x y a x como la salida esperada de y. Por lo tanto estas redes tienen un mecanismo de aprendizaje híbrido: una parte es no supervisada para abstraer las características de los vectores x, como en los mapas auto-organizativos, y la otra parte es supervisada ya que se entrena utilizando como salida esperada los vectores y. Una ventaja de la red de contra-propagación es que el entrenamiento es más rápido que el de una red *backpropagation*, en la que el error de cada neurona de una capa se calcula a partir del error de todas las neuronas de la capa siguiente, pero tiene la desventaja de que la precisión alcanzada no es tan alta como la de la red *backpropagation*.

Las neuronas de la capa oculta forman una red competitiva que se entrena de la misma forma que un mapa auto-organizativo: cada ejemplo es presentado a la red, se designa la neurona ganadora, y se actualizan los pesos de dicha neurona, como se indica en la ecuación 2.7. La diferencia con los mapas auto-organizativos es que no se utiliza el concepto de vecindario, la única neurona que se actualiza es la ganadora.

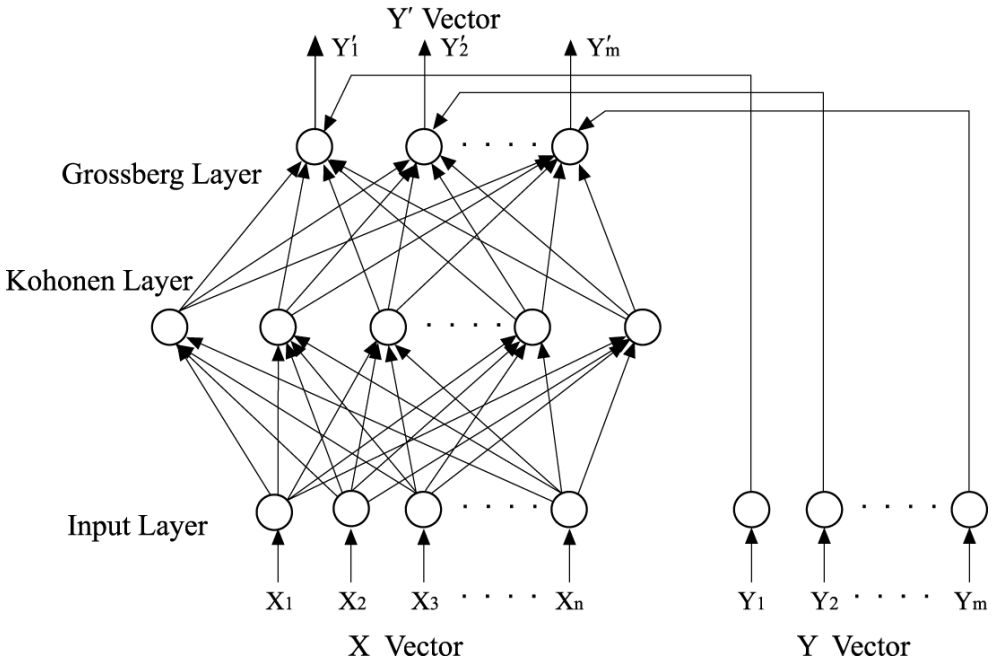

Figura 2.7: Red de contra-propagación (CPN)

Como se observa en la Figura 2.7 todas las entradas se conectan con las neuronas ocultas, y la salida de todas las neuronas ocultas se conecta con la entrada de cada neurona de la capa de salida. En la figura se representa la salida de la red con Y', la cual es comparada contra el vector Y para determinar el error. El entrenamiento de la capa de salida se lleva

a cabo de forma similar al de la capa oculta, utilizando una medida de distancia, pero el objetivo en este caso no es agrupar los ejemplos de entrada en regiones sino encontrar una asociación entre estas regiones y los vectores y. De ahí viene el nombre de las redes de contra-propagación, del hecho que los ejemplos de entrada ingresan a la red desde ambos extremos.

El entrenamiento consiste en ingresar un vector x a la red, determinar la neurona oculta ganadora j que esté más cerca de x, y actualizar los pesos que *salen* de la ganadora hacia las unidades ocultas. La ecuación 2.8 muestra cómo se actualizan los pesos de las neuronas de la capa de salida, indicados con v para distinguirlos de los pesos de las neuronas de las capas ocultas.

De esta forma en los pesos de la capa de salida quedan representados los vectores y que se corresponden con grupos de vectores x que son cercanos en el espacio n-dimensional de entrada. Cada uno de estos grupos de vectores x es representado por una neurona competitiva de la capa oculta. En la capa de salida se utiliza un factor de aprendizaje β distinto al utilizado en la capa oculta, permitiendo que las capas puedan ser entrenadas y ajustadas independientemente para alcanzar los mejores resultados.

$$\forall o : v_{oj(t+1)} = v_{oj(t)} + \beta(y_j - v_{oj}) \tag{2.8}$$

2.4. Algoritmo K-medias

En esta sección se describe el método de agrupamiento K-medias [30]. Este método obtiene K regiones o grupos a partir de los datos de entrada en torno a K centros, que pueden o no coincidir con instancias de los datos de entrada. Como se describió en la Sección 2.2, ésta es una técnica basada en casos y vecindad ya que trabaja con el concepto de *distancia* entre ejemplos. Esto hace que sea necesario definir la medida de distancia a utilizar ya que la calidad de los grupos encontrados, medida por la cohesión de los mismos, depende fuertemente de la medida utilizada en un conjunto de datos particular. Como el propósito de K-Medias es encontrar grupos cuyos elementos sean similares se necesita definir formalmente la similitud entre elementos y para ello se utiliza el concepto de distancia, de forma que la similitud entre los elementos sea alta cuando la distancia entre ellos sea baja.

Al principio del algoritmo los K centros son elegidos aleatoriamente entre los ejemplos de entrada. Los K grupos iniciales se forman asociando cada ejemplo con el centro más cercano. A medida que el algoritmo se ejecuta los ejemplos cambian de grupo, los centros deben calcularse nuevamente mediante la media n-dimensional de los ejemplos

que pertenecen al grupo. El objetivo del algoritmo es minimizar las distancias de los ejemplos a los centros de sus respectivos grupos.

Además de los ejemplos de entrada, el algoritmo recibe como parámetro el valor K que designa la cantidad de grupos que se quiere identificar, y produce como salida los K centros encontrados y la asignación de cada ejemplo de entrada a uno de los grupos encontrados. La fase de agrupamiento es lenta, pero una vez obtenidos los centros la clasificación de un ejemplo nuevo se realiza rápidamente ya que consiste solamente en calcular la distancia del mismo a cada uno de los K centros y elegir el más cercano.

Formalmente el algoritmo trabaja de la siguiente manera:

- Dado un ejemplo x, se identifica el centro más cercano A_k y se incluye el ejemplo en la lista de los ejemplos pertenecientes a dicho centro.

$$A_k = argmin_{A_i}\{dist(x, A_i)\}, i = 1..K$$

- Después de que se hayan analizado todos los ejemplos, cada centro A_k cuenta con un conjunto de m_k ejemplos a los que representa:

$$I(A_k) = \{x_{k1}, x_{k2}, \ldots, x_{km_k}\}$$

- Se recalculan los centros, desplazándolos hacia el centro del conjunto de ejemplos correspondiente:

$$A_k = \frac{\sum_{i=1}^{m_k} x_{ki}}{m_k}$$

- Se repite el procedimiento hasta que ya no se muevan los centros.

El algoritmo de K-medias es similar a las redes SOM de Kohonen en la noción de desplazar los centros para minimizar las distancias a los ejemplos. La diferencia principal es que en los SOM los centros se mueven con cada ejemplo analizado mientras que en K-medias los centros se desplazan una vez que se analizaron todos los ejemplos en una iteración dada. A medida que los centros se acomodan se producen menos cambios en sus conjuntos de ejemplos, y por consiguiente el desplazamiento de cada centro es menor. Los desplazamientos son cada vez menores hasta que eventualmente cesan.

El algoritmo de K-medias es simple y fácil de aplicar. Sin embargo, una de las cuestiones más importantes a la hora de aplicarlo es determinar el valor de K. Si el valor de K es

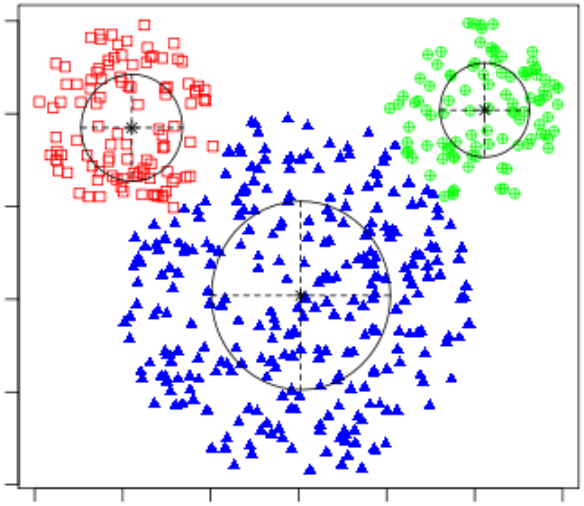

Figura 2.8: K-medias

muy bajo, se obtendrán grupos muy grandes formados por elementos poco similares entre sí. Contrariamente, si el valor de K es muy alto se obtendrán muchos grupos reducidos cuyos centros son similares. Ninguna de estas situaciones es deseable ya que la calidad de los grupos formados se mide por la cohesión de los mismos, la cual consiste en que los elementos que forman cada grupo sean similares entre sí y diferentes de los elementos de otros grupos. La aleatoriedad inicial de los centros también condiciona el rendimiento del algoritmo, ya que es posible que no cubran uniformemente el espacio de entrada con lo que se hace difícil hallar un centro que represente adecuadamente a cada grupo existente en los datos de entrada.

2.5. Resumen del capítulo

En este capítulo se describieron formalmente las tareas más comunes de la minería de datos y las técnicas habitualmente utilizadas para resolverlas. Las tareas se clasifican en tareas predictivas, cuando la finalidad de las mismas es predecir valores desconocidos sobre datos nuevos, y descriptivas, cuando la finalidad es describir y caracterizar los datos existentes. Entre las técnicas de la minería de datos más populares se encuentran los árboles de decisión, los sistemas de reglas, las redes neuronales, las técnicas de modelización estadística, las técnicas bayesianas, y las técnicas basadas en distancias y vecindad, entre otras.

Cada una de estas técnicas está orientada a resolver cierto tipo de tarea y tiene diferentes

características en cuanto a la precisión, eficiencia, expresividad, comprensibilidad y facilidad de uso. La decisión de qué herramienta utilizar ante un problema dado no siempre es clara y a menudo se requiere un proceso de prueba y error para encontrar el método más adecuado al problema que se quiere resolver.

Dentro de las técnicas mencionadas se dio un especial énfasis a las utilizadas en el desarrollo del método de caracterización de documentos presentado en este trabajo, propiamente las redes neuronales y el algoritmo de K-medias. Las redes neuronales tienen una gran variedad de aplicaciones y son aptas para encontrar relaciones y características generales a partir de los datos de entrenamiento. Esto permite capturar ejemplos que no son evidentes o fáciles de expresar o transmitir, como por ejemplo las características que definen una "buena" palabra clave dentro de un documento. Dentro de las redes neuronales se distinguen las topologías y algoritmos de entrenamiento que siguen los modelos de aprendizaje supervisado y no supervisado, y se describen las características principales de las redes más representativas de cada tipo de aprendizaje.

Teniendo en cuenta los conceptos, nociones y definiciones presentadas sobre el proceso de la minería de datos y las técnicas que la misma engloba, estamos en condiciones de profundizar en la descripción de LIKE, el método de caracterización de documentos mediante la extracción de palabras clave que es el tema central del presente trabajo.

Capítulo 3

El método LIKE. Pre-procesamiento de los datos

El tema central de esta tesina de grado es el desarrollo de un método de extracción de palabras clave a partir de un conjunto de documentos. El enfoque utilizado fue el de convertir el problema de la extracción de palabras clave en un problema de clasificación, de forma que el método sea capaz de determinar para cada término presente en el documento si el mismo es una palabra clave o no.

El método propuesto en este trabajo se llama LIKE (Language Independent Keyword Extraction), y es un método de aprendizaje supervisado que construye un modelo de clasificación a partir de un conjunto de documentos que utiliza en una etapa de entrenamiento. Como su nombre lo indica, LIKE no depende de características lingüísticas particulares del lenguaje a tratar como otras aproximaciones que utilizan diccionarios o listas de palabras a excluir (*stopwords*). Sin embargo, LIKE necesita una gran cantidad de documentos de entrenamiento que pertenezcan al dominio en cuestión (ámbito científico, noticias, etc.) para obtener un clasificador que funcione adecuadamente. Es importante que los documentos utilizados en el entrenamiento pertenezcan al mismo dominio y que los documentos a clasificar también sean de este dominio ya que LIKE busca extrapolar las características de la estructura implícita del texto, como la organización y la distribución de las palabras importantes y el estilo de redacción empleado. Si los documentos no pertenecen al mismo dominio es muy probable que estas similitudes estructurales no existan, además de que el vocabulario usado en cada uno de ellos podría ser radicalmente diferente del resto, lo que perjudica el entrenamiento y por lo tanto en la precisión de LIKE.

A grandes rasgos, LIKE transforma los documentos para obtener una serie de características descriptivas que sirvan para clasificar los términos. Así, cada término se transforma

en un vector de características numéricas y estos vectores son utilizados para entrenar una red neuronal *backpropagation*. El entrenamiento supervisado utiliza como la información de la clase aquellos términos elegidos como palabras clave por los autores de los documentos. Una vez entrenada la red, ésta se puede utilizar para clasificar términos provenientes de documentos nuevos. La estructura de LIKE se visualiza en la figura 3.1.

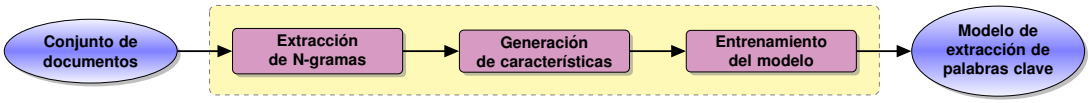

Figura 3.1: LIKE. Obtención del modelo de clasificación

Este capítulo está organizado de la siguiente manera. En la Sección 3.1 se analiza en detalle el problema de la extracción de palabras clave, y se discuten las diferentes propuestas que han surgido para resolver este problema. En la Sección 3.2 se presenta la estructura general de LIKE y se describe la etapa de limpieza y preparación de los datos. En la Sección 3.3 se detalla la etapa más importante de LIKE que es la transformación del texto que genera las características de los términos que son utilizadas en la etapa de entrenamiento del modelo de clasificación.

3.1. El problema de la extracción de palabras clave

Como se comentó en la Sección 1.4 el problema de la extracción de palabras clave puede ser visto como un problema de categorización de textos, con el fin de caracterizar los documentos y agruparlos en conjuntos de textos que hablan de los mismos temas. El uso de palabras clave facilita la búsqueda y recuperación de la información ya que las mismas describen de forma breve y precisa el contenido de un documento. Esto permite que un usuario pueda determinar si undocumento es de su interés sin tener que leerlo, basándose en las palabras clave. Sin embargo, no todos los documentos tienen palabras clave asociadas y el proceso de asignarlas es lento y arduo. Es por esto que surge la necesidad de contar con herramientas que asistan a los profesionales en la realización de esta tarea.

Hay dos formas de tratar la identificación de palabras clave, la asignación *(keyword assignment)* y la extracción propiamente dicha *(keyword extraction)*.

En la *asignación* de palabras clave se trabaja con un vocabulario controlado del cual provienen los términos a ser utilizados para caracterizar los documentos. Cualquier término que no aparezca en este vocabulario no puede ser elegido como palabra clave. La

asignación de palabras clave también se conoce como *text indexing*, como se mencionó en la Sección 1.4.2. Las ventajas principales de este enfoque son la simplicidad y la consistencia. La simplicidad radica en que es sencillo considerar al problema como un problema de clasificación en el que cada término del vocabulario es una clase. La consistencia, por otro lado, se debe a que al utilizar una cantidad acotada de términos se asegura que los documentos similares sean descriptos por las mismas palabras clave. Sin embargo, las desventajas principales de este enfoque consisten en que el vocabulario es difícil de crear y mantener (como ocurre con las bases de conocimiento), lo que hace que no estén disponibles siempre, y además en que puede haber términos presentes en los documentos que pueden resultar útiles para describirlos pero que no pueden ser elegidos porque no forman parte del vocabulario controlado.

Por otro lado, en la *extracción* propiamente dicha las palabras clave son elegidas exclusivamente entre los términos que aparecen en el texto. Estos términos usualmente son extraídos utilizando algún mecanismo de aprendizaje automático que trabaja a partir de un conjunto de características de los términos, el cual es el enfoque empleado en este trabajo. La ventajas de este enfoque radican en la flexibilidad, ya que cualquier término representativo puede ser elegido, y en que no es necesario usar un vocabulario controlado. La desventaja principal es la pérdida de la consistencia, ya que estos métodos son afectados por la sinonimia (términos distintos pueden referir al mismo concepto), y la dificultad en elegir las palabras debido a que la cantidad de términos elegibles no está acotada.

Como se comentó también en la Sección 1.4, un enfoque muy popular es tratar este problema como un problema de clasificación utilizando técnicas de aprendizaje automático. Como los clasificadores no pueden trabajar con el texto en su forma original, es necesario proporcionarle al mismo una representación adecuada para las tareas de aprendizaje. Comúnmente en el aprendizaje automático los documentos son representados mediante vectores de características, siendo la representación más sencilla la de *bag-of-words*. Esta representación utiliza todas las palabras, con lo que la dimensión del vector es igual a la cantidad total de palabras en todos los documentos analizados. Hay diferentes métodos para dar peso a las características contenidas en los vectores. El más simple consiste en asignar un valor binario que indica si la palabra está presente o no en el documento. Los esquemas más complejos en cambio consideran la frecuencia de las palabras dentro del texto, y dentro de estos esquemas el más común es el TF-IDF [39], definido en la ecuación 3.2 y explicado en la Sección 3.3.2.

Uno de los primeros avances en considerar el problema de la extracción de palabras clave como un problema de clasificación a resolver mediante el aprendizaje automático es el de Peter Turney [41]. Los estudios de Turney afirman que el 70 % de las palabras clave

elegidas por el autor aparecen dentro del documento, lo que hace factible utilizarlas en técnicas de aprendizaje supervisado. Turney desarrolló un algoritmo que llamó GenEx, en el cual se utiliza una serie de reglas cuyos parámetros son afinados en una primera etapa mediante un algoritmo genético. Estas reglas son utilizadas para calificar los términos y seleccionar aquéllos con el mayor puntaje en una segunda etapa. GenEx utiliza términos formados por hasta tres palabras consecutivas y para cada uno de ellos calcula varios atributos, entre ellos la frecuencia *(Term Frecuency, TF)*, la primera aparición dentro del texto y la longitud del término medida en cantidad de palabras. GenEx tiene una fase de pre-procesamiento en el que se aplica *stemming* a los términos, se filtran *stop-words*, y se extraen los 1-gramas, 2-gramas y 3-gramas.

Otro ejemplo representativo es KEA, desarrollado por Ian Witten y Eibe Frank [15]. KEA utiliza como características de los términos el TF-IDF y la posición dentro del texto, y también se menciona la incorporación de un tercer atributo que representa la cantidad de veces que un término dado fue seleccionado como palabra clave en el conjunto de documentos, con lo que es necesario que los documentos pertenezcan al mismo dominio para que este atributo tenga significado. Este algoritmo utiliza un clasificador Naive Bayes que calcula para cada término la probabilidad de ser elegido como palabra clave. KEA también aplica *stemming* y filtrado de *stop-words*. Los estudios de Frank y Witten afirman que si bien la *performance* de KEA es comparable a la de GenEx, el entrenamiento es más simple y eficiente. KEA es de particular interés en este trabajo ya que es el algoritmo de referencia contra el que se compara la precisión de LIKE.

Anette Hulth desarrolló un sistema de extracción de palabras clave a partir de *abstracts* de artículos científicos [24]. Hulth remarca la importancia de considerar aspectos lingüísticos como la categoría gramatical de una palabra en la representación de los términos del documento, y hace énfasis en esta representación. Hulth realizó experimentos utilizando tres representaciones distintas: n-gramas, frases sustantivas *(noun phrases)* y patrones del discurso *(part-of-speech (POS) patterns)*. A los atributos utilizados por Witten y Frank se agregó la categoría de la palabra, sea ésta un sustantivo, adjetivo, verbo, etc. Como técnica de aprendizaje automático Hulth utiliza el sistema *Rule Discovery Sistem (RDS)*, que permite extraer reglas en forma de un árbol de decisión.

Yaakov HaCohen-Kerner, Zuriel Gross y Asaf Masa [19] desarrollaron un sistema que utiliza un alto número de características y utiliza esta representación para entrenar diferentes modelos de aprendizaje automático y combinar sus resultados. Los atributos de los términos utilizados son entre otros TF-IDF, primera aparición, posición dentro del párrafo y semejanza con el título. Los autores utilizaron para sus experimentos implementaciones del árbol de decisión C4.5, del perceptrón multicapa, y de Naive Bayes. Algunos de estos atributos son utilizados también en LIKE.

Sin embargo no todos los enfoques son de aprendizaje supervisado.

Rada Mihalcea y Paul Tarau [33] proponen TextRank, un algoritmo no supervisado de extracción de palabras clave. TextRank es una adaptación de PageRank [36], el algoritmo de Lawrence Page que se utiliza para medir la importancia de las páginas web en base a los links que las referencian. TextRank trabaja sobre una representación basada en grafos con pesos y calcula para cada nodo un puntaje que depende del puntaje de sus nodos vecinos. En TextRank los nodos representan términos y las aristas entre ellos representan relaciones entre los términos. Estas relaciones vienen dadas por la co-ocurrencia de ambos términos en una ventana deslizante de tamaño fijo. Cuando el algoritmo converge y los puntajes de los nodos se estabilizan TextRank selecciona los términos con el puntaje más alto. TextRank aplica filtros sintácticos para seleccionar términos formados por sustantivos y adjetivos, siguiendo la noción de que las palabras clave normalmente están formadas por este tipo de palabras. TextRank también puede aplicarse a tareas de resumen automático mediante la extracción de oraciones significativas dentro del texto. Una de las ventajas de TextRank es que no tiene fase de entrenamiento y se puede aplicar a documentos individuales sin la necesidad de contar con un conjunto de documentos.

Otro algoritmo no supervisado que trabaja sobre documentos individuales es el de Yutaka Matsuo y Mitsuru Ishizuka [31]. Ellos utilizan una medida chi-cuadrada (χ^2) para calcular la distribución de probabilidad de la co-ocurrencia de los términos del documento con un conjunto seleccionado de los términos más frecuentes dentro del mismo documento. Esta medida determina la importancia del término y es utilizada para ordenar los términos y seleccionarlos en base a su puntaje. Este método tiene una fase de pre-procesamiento en el que se extraen los términos, se filtran *stop-words*, y se aplica stemming para unificar palabras con la misma raíz sintáctica.

Como se observa, hay muchos enfoques distintos para resolver el problema de la extracción de palabras clave. Varios de éstos métodos son supervisados y requieren de un conjunto de datos de entrenamiento. Otros en su lugar buscan hallar estructuras y relaciones dentro del texto mismo. Algunos métodos más avanzados recurren a bases de conocimiento externo con el fin de brindar mayor contenido semántico a la representación de los términos. En el presente trabajo se optó por el enfoque del aprendizaje supervisado ya que es uno de los más representativos y consolidados dentro del área de la minería de textos. Varios de los atributos utilizados en las propuestas de los autores mencionados son también incluidos en LIKE, como TF, TF-IDF y la posición de la primera aparición del término. Estos atributos permiten caracterizar en cierta forma el uso y la función lingüística que cumplen los términos más allá de su significado concreto.

Una cosa importante a destacar es que, a diferencia de varios de los métodos mencionados, en LIKE se decidió no utilizar *stemming* y *stop-lists*. Hay varias razones para esta

decisión.

La primera de ellas es que el propósito de LIKE es la independencia del lenguaje utilizado en el documento. Dicha independencia se perdería al utilizar *stop-lists*, ya que son listas compuestas por palabras conectoras y de poco significado que naturalmente dependen del lenguaje particular. El *stemming* por su parte involucra un conjunto de sufijos y prefijos sintácticos que denotan modificaciones de género, número, cualidades, etc. a la palabra que están modificando, y por lo tanto también son dependientes del lenguaje en cuestión.

Otra razón es la intención de capturar palabras clave representadas por términos compuestos, como "ingeniería de software", que contienen conectores y que se perderían si se aplicara una *stop-list*, o haría necesario aplicar un post-procesamiento que reconstruya términos compuestos a partir de componentes más simples.

La tercera razón es que estos mecanismos están muy bien desarrollados para el idioma inglés pero no así para el español y otros idiomas, con lo que la independencia del lenguaje se vería aún más perjudicada. Además estas listas y procedimientos de eliminación de prefijos y sufijos necesitan ser desarrolladas bajo la supervisión de lingüistas o profesionales del lenguaje que verifiquen su corrección.

3.2. LIKE: Recopilación, integración y limpieza de los datos

En esta sección se describe el proceso de adquisición y preparación de los documentos utilizados para el desarrollo, entrenamiento y evaluación de LIKE. En la sección siguiente se detallará el proceso de transformación de los textos que permite obtener la representación utilizada por el algoritmo de aprendizaje.

Los textos utilizados para los experimentos realizados provienen de una base de datos proporcionada por el Servicio de Difusión de la Creación Intelectual (SeDiCI) [5], el repositorio institucional de la Universidad Nacional de La Plata cuyo propósito es brindar acceso libre y gratuito a la producción científica de la Universidad al público en general. Esta información es recopilada y organizada por el personal del Proyecto de Enlace de Bibliotecas (PrEBi) [4] de la Universidad.

La base de datos está compuesta por artículos científicos en español y en inglés publicados en el Workshop de Investigadores en Ciencias de la Computación (WICC) [6] desde el año 1999 hasta el año 2012. De cada artículo se utilizó el título, el *abstract* y el resumen. Esta base de datos también incluye las palabras clave de cada artículo indicadas por sus autores. Estas últimas fueron utilizadas en la etapa de entrenamiento supervisado para

designar la clase (“es palabra clave” / “no es palabra clave”) a la que pertenecen los términos y también para evaluar la precisión en la etapa de evaluación del método.

En esta etapa se ignoraron los artículos sin palabras clave ya que no pueden ser utilizados para la fase de entrenamiento. También se uniformaron los datos pasando todas las palabras a minúscula, y se unificó la puntuación de siglas y términos compuestos separados por guiones para que el cálculo de los errores y los aciertos en las predicciones del modelo se realice correctamente.

La primera etapa de LIKE consiste en la generación de los vectores de características de los términos que se van a utilizar en el entrenamiento. Para ello se utilizan los documentos en texto libre y un archivo adicional que contiene meta-información de los documentos, más precisamente el título y las palabras clave de cada uno. El pre-procesamiento aplicado a los documentos consiste también en quitar el título y la bibliografía, eliminar saltos de línea y tabulaciones y reemplazar múltiples blancos por blancos individuales.

3.3. Transformación de los datos

Una vez que se llevó a cabo el pre-procesamiento de los textos LIKE procede a extraer los términos. Como se comentó en la Sección 1.4, la representación elegida para los términos del texto es la de n-gramas, los cuales consisten en secuencias de *n* palabras consecutivas dentro de una oración.

LIKE recibe como parámetros dos listas formadas por separadores de palabras y separadores de oraciones. Esto permite configurar la forma en que LIKE extrae los términos de los textos, ya que en determinados dominios o aplicaciones podría ser deseable obtener términos compuestos que contengan caracteres que en otros casos serían considerados separadores. Por ejemplo, en el dominio informático podríamos querer designar como un separador de oraciones al *string* “. ” (punto y blanco) en lugar del punto para permitir que haya palabras que contengan puntos, como nombres de tecnologías (como podría ser “Web 2.0”) o incluso URLs.

Los separadores empleados son generalmente signos de puntuación como puntos, comas, paréntesis y guiones, pero podrían utilizarse combinaciones arbitrarias de símbolos como separadores. Los caracteres designados como separadores no van a formar parte de las palabras extraídas de los textos, y por lo tanto tampoco van a estar en las palabras clave. Usando estos separadores, LIKE procede a dividir el texto en oraciones y a cada oración en *tokens* o palabras individuales. La decisión de considerar oraciones como unidades semánticas está motivada por la presencia de ciertos atributos de los términos que resultan de interés como por ejemplo la posición relativa dentro de una oración. Además se evita

que dos *tokens* de oraciones distintas sean asociados en el mismo n-grama.

Una vez dividido el texto en oraciones y palabras empieza la etapa de extraer los términos y sus características.

3.3.1. Obtención de los términos

Otro parámetro que tiene LIKE es el valor de *n*, que representa la cantidad máxima de palabras individuales que pueden formar parte de un término. Hay estudios que afirman que en el ámbito científico comúnmente las palabras clave tienen entre 1 y 3 palabras [17, 35, 40]. Sin embargo, también hay dominios específicos en los se suelen emplear palabras clave más largas. Por esta razón la cantidad máxima de palabras de los términos es un parámetro de LIKE, para flexibilizar la extracción de los términos y ajustarla acorde al dominio de aplicación. Un valor alto de *n* resulta en que se extraiga mayor cantidad de términos, con lo que la construcción del modelo de clasificación tarda más tiempo y exige más recursos. Por el contrario, un valor bajo de *n* acelera el entrenamiento, pero reduce la cantidad de términos que se pueden encontrar y clasificar como palabras clave.

Una primera aproximación es generar todos los posibles n-gramas dentro de cada oración. Si bien esta aproximación es sencilla y a simple vista parece lógica, para textos largos se genera una cantidad excesivamente grande de n-gramas para procesar, muchos de los cuales podrían descartarse antes del procesamiento ya que no todas las combinaciones de palabras consecutivas dan lugar a términos con significado o con relevancia al tema tratado en el documento.

Hay muchos criterios intuitivos para determinar si una palabra es importante dentro de un documento y contribuye a describir su contenido. Algunos de estos criterios pueden ser subjetivos y dependientes de cada autor. Es por esta razón que en este trabajo se decidió adoptar un criterio estadístico para abstraer y evaluar las características de los términos que hacen que los distintos autores los elijan como palabras clave.

El criterio más sencillo es considerar como palabra clave a un término que aparece muchas veces en el texto. Si bien parece tener sentido que una palabra importante tenga una frecuencia alta en el texto, hay palabras muy frecuentes con bajo contenido semántico como preposiciones y artículos cuya función es precisamente relacionar y conectar conceptos importantes y facilitar el desarrollo del discurso y de las ideas plasmadas en el texto. Estas palabras tienen una frecuencia muy alta dentro de un documento ya que se utilizan para estructurar el hilo de pensamiento del autor, pero no son palabras clave dado que no expresan conceptos concretos por sí mismas. Por otro, pueden haber documentos con palabras clave que tengan frecuencia baja en el texto o que no aparezcan en absoluto.

Esto puede ocurrir especialmente con términos muy generales que dan marco a las ideas específicas presentadas en el texto. Sin embargo, si bien la frecuencia puede no ser por sí sola un indicador de la importancia de un término, muchos métodos de extracción de palabras clave la utilizan como uno de los atributos o características a tener en cuenta para su posible selección.

Así como los términos muy frecuentes pueden resultar poco significativos, lo mismo ocurre con los términos poco frecuentes. Su eliminación antes de comenzar el procesamiento tiene el propósito de reducir la dimensión del problema, contribuyendo a la eficiencia en tiempo y en recursos. Por esta razón LIKE contiene en esta etapa una implementación del algoritmo desarrollado por Johannes Fürnkranz [17] para extraer eficientemente los n-gramas que tengan una frecuencia mínima especificada sin la necesidad de generar todos los posibles n-gramas. El algoritmo de Fürnkranz es similar al algoritmo Apriori de extracción de reglas de asociación en el aspecto de que ambos construyen conjuntos de elementos a partir de subconjuntos menores que cumplan con un criterio dado.

Este algoritmo recibe como parámetros el valor *n* que representa la longitud máxima de los n-gramas a extraer y la frecuencia mínima que cada término debe tener para ser extraído. La idea es generar n-gramas a partir de (n-1)-gramas que cumplan con la condición de tener la frecuencia mínima especificada. Para ello se utiliza la propiedad de que un n-grama con frecuencia *k* se construye a partir de la intersección de dos (n-1)-gramas con frecuencia al menos *k*. En otras palabras, esta propiedad afirma que un n-grama no puede ser más frecuente que las partes que lo componen, con lo que se pueden filtrar n-gramas infrecuentes sin la necesidad de contabilizar sus ocurrencias si ya se sabe que sus partes son infrecuentes.

El algoritmo realiza *n* pasadas sobre el texto. En una primera etapa se obtienen los 1-gramas (palabras individuales) que tengan la frecuencia mínima especificada y partir de ellos se obtienen los 2-gramas. Luego se procede de la misma forma para obtener los 3-gramas, y así sucesivamente. Para cada n-grama se toman sus primeras y sus últimas *n-1* palabras para verificar que ambos (n-1)-gramas cumplan con el criterio de frecuencia mínima. Si este criterio no se cumple, el n-grama en cuestión es descartado ya que su frecuencia no será mayor que la de sus partes. Al finalizar cada pasada se descartan también los n-gramas extraídos que no tengan la frecuencia mínima deseada y se comienza con la pasada siguiente.

3.3.2. Generación de características

Una vez que se tienen los términos en forma de n-gramas se pueden calcular los atributos que los caracterizan y que van a ser utilizados en el entrenamiento. Si bien conceptualmente las fases de separar el texto en palabras y oraciones, extraer los n-gramas filtrando aquéllos que son infrecuentes, y generar las características de los n-gramas son secuenciales, en la implementación presentada se realizan simultáneamente por razones de eficiencia.

Los n-gramas también tienen otros atributos asociados como el *string* literal del término representado, la longitud del n-grama y el documento del que fue extraído. Estos atributos adicionales describen el término y facilitan la evaluación y la visualización de los resultados pero no son utilizados en el entrenamiento del algoritmo. Los n-gramas iguales de documentos diferentes son considerados instancias diferentes en el entrenamiento, ya que las características calculadas no necesariamente son iguales.

Las características de los n-gramas calculadas a partir del texto son:

1. Frecuencia del término *(Term Frequency, TF)*: es el indicador más utilizado para caracterizar documentos. Se calcula dividiendo la frecuencia del término por la cantidad total de palabras del texto. Sea t un término y d un documento:

$$TF(t,d) = \frac{freq(t,d)}{length(d)} \tag{3.1}$$

   donde *freq(t,d)* es la cantidad de veces que t aparece en d y *length(d)* es la cantidad de palabras del documento d.

2. TF-IDF *(Term Frequency - Inverse Document Frequency)*: consiste en ponderar el TF de un término con la frecuencia que tiene dicho término en el cuerpo entero de documentos. IDF representa qué tan infrecuente es un término en el cuerpo de documentos mediante la relación entre la cantidad de documentos que lo contienen y la cantidad total de documentos. Esto se basa en el criterio de que un término que caracteriza un documento y lo distingue de otros documentos es frecuente en ese documento y poco frecuente en el resto. Así, IDF toma valores bajos para términos que aparecen en muchos documentos y valores altos para términos que aparecen en pocos. IDF toma el valor cero para términos que aparecen en todos los documentos.

   IDF se define como:

$$IDF(t,D) = log\left(\frac{|D|}{|d \in D : t \in d|}\right)$$

   donde D es el cuerpo entero de documentos, $|D|$ es la cantidad de documentos del cuerpo, y $|d \in D : t \in d|$ es la cantidad de documentos que contienen el término t.

Luego, TF-IDF se define como:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D) \tag{3.2}$$

De esta forma IDF balancea el valor de TF favoreciendo a los términos que aparecen en pocos documentos y son frecuentes en dichos documentos. TF-IDF es también junto con TF uno de los indicadores más utilizados en los métodos de minería de textos.

3. Primera ocurrencia de un término *(firstOcurrence)*: representa la posición relativa de la primera ocurrencia del término dentro del texto. Se calcula como la cantidad de palabras individuales antes de la primera aparición del término dividido por la cantidad de palabras del documento. Este atributo simboliza la noción intuitiva de que generalmente los conceptos importantes del documento se presentan en la introducción de un documento y son resumidos al final del mismo, con lo que la posición inicial del término también influye en lo que constituye una palabra clave.

$$firstOcurrence(t, d) = \frac{pos(t, d)}{length(d)} \tag{3.3}$$

donde *pos(t,d)* es la posición que ocupa el término t en d. Si t es un término formado por varias palabras, *pos(t,d)* es la posición de su primera palabra.

4. Posición dentro de la oración *(positionInSentence)*: es una medida de la posición relativa que ocupa el término dentro de las oraciones en las que aparece. Para cada oración s en la que t aparece, se calcula la cantidad de palabras de s antes de la aparición de t. El resultado es entonces el promedio de estos valores. Puede ocurrir que t aparezca varias veces en la misma oración, en ese caso cada ocurrencia es tratada por separado de la misma forma que si hubieran ocurrido en oraciones distintas. Si consideramos a la oración s como un conjunto de palabras, podemos verla como un subconjunto del documento d:

$$positionInSentence(t, d) = \frac{\sum_{s \subset d : t \in s} \frac{pos(t, s)}{length(s)}}{freq(t, d)} \tag{3.4}$$

donde *pos(t,s)* indica la posición del término t en la oración s. En la ecuación anterior, la expresión $s \subset d : t \in s$ hace referencia a las oraciones del documento d en las que aparece el término t. Aunque se adopta esta notación por simplicidad, es importante destacar que las oraciones no son conjuntos en el sentido estricto ya que sus elementos tienen orden y son contiguos. Además, no cualquier subconjunto de palabras del documento es considerado una oración.

5. Aparición en el título *(isInTitle)*: este atributo vale 1 si el término aparece textualmente en el título del documento y 0 si no aparece. Representa la noción de que las palabras que aparecen en el título de un documento son importantes y por ende son candidatas a ser palabras clave.

$$isInTitle(t,d) = \begin{cases} 1 & \text{si } t \in title_d \\ 0 & \text{si } t \notin title_d \end{cases} \tag{3.5}$$

donde $title_d$ es el título del documento d. Siguiendo la convención del punto anterior, como el título es una oración se lo considera un conjunto para facilitar la notación.

6. Aparición de los miembros en el título *(membersInTitle)*: este atributo, al igual que el anterior, relaciona la importancia de un término con su aparición en el título. La diferencia está en que se considera la ocurrencia en el título de las palabras individuales que componen el término. Esto permite considerar a los términos cuyas ocurrencias en el título no son textuales, como cuando las palabras están en un orden distinto o se quitan o agregan términos conectores. Se calcula contando cuántas palabras individuales del término t aparecen en el título del documento y dividiendo esta cantidad por la longitud de t.

$$membersInTitle(t,d) = \frac{\sum_{w \in t} isInTitle(w,d)}{length(t)} \tag{3.6}$$

donde w es una palabra individual y el término t es considerado un conjunto de palabras individuales.

7. Longitud normalizada de la oración *(NormalizedSentenceLength, NSL)*: es una medida de la longitud de las oraciones en las que aparece un término. Se calcula promediando la longitud normalizada de las oraciones en las que un término aparece. La longitud de una oración se normaliza dividiéndola por la longitud de la oración más larga. [26]

$$NSL(t,d) = \frac{\sum_{s \subset d : t \in s} \frac{length(s)}{max_{s' \subset d}\{length(s')\}}}{freq(t,d)} \tag{3.7}$$

8. Frecuencia normalizada *(Z-Score)* [8]: consiste en normalizar la frecuencia del término usando la frecuencia media en el cuerpo de documentos y la desviación estándar de dicha frecuencia.

$$Z\text{-}Score(t, d, D) = \frac{freq(t, d) - \mu(t, D)}{\sigma(t, D)} \tag{3.8}$$

donde

$$\mu(t, D) = \frac{\sum_{d \in D} freq(t, d)}{|D|}$$

y

$$\sigma(t, D) = \sqrt{\frac{\sum_{d \in D} (freq(t, d) - \mu(t, D))^2}{|D| - 1}}$$

En las ecuaciones anteriores $\mu(t, D)$ representa la media muestral de la frecuencia de t en el cuerpo de documentos D y $\sigma(t, D)$ representa la desviación estándar muestral de la frecuencia.

9. Condición de ser palabra clave elegida por el autor *(isKeyword)*: este atributo funciona como la etiqueta de clase de los ejemplos usados en el entrenamiento del modelo de clasificación. A diferencia de los anteriores, el valor de este atributo no necesita ser calculado sino que es extraído directamente de los meta-datos que acompañan cada documento. Este es el atributo cuyo valor hay que determinar a partir de las características de los términos de un documento nuevo. En otras palabras, es el atributo *clase* de la instancia.

$$isKeyword(t, d) = \begin{cases} 1 & \text{si } t \in keywords_d \\ 0 & \text{si } t \notin keywords_d \end{cases} \tag{3.9}$$

donde $keywords_d$ es el conjunto de palabras clave del documento d indicadas por el autor.

De las 9 características utilizadas TF-IDF y Z-Score dependen del cuerpo entero de documentos para su cálculo. Además éstos son los únicos atributos cuyos valores no están entre 0 y 1. Los demás atributos se encuentran por definición entre 0 y 1, con el fin de unificar y acotar los rangos de valores que pueden tomar.De esta forma se conservan las relaciones de los valores originales y se facilita el proceso de aprendizaje.

El atributo Z-Score mide a cuántas desviaciones de distancia está la frecuencia del término respecto a la frecuencia promedio del mismo término en todos los documentos del cuerpo, por lo que puede tomar valores por fuera del rango entre 0 y 1 e incluso valores negativos. Por su parte TF-IDF pondera la frecuencia de un término respecto a su frecuencia en el cuerpo de documentos y puede tomar valores más altos que 1 para aquéllos términos que son frecuentes dentro de pocos documentos. Por esta razón es conveniente aplicar un

segundo escalado a estos datos a la hora de entrenar la red neuronal en la etapa siguiente de LIKE.

3.4. Resumen del capítulo

En este capítulo se dio una descripción general de LIKE, el método de extracción de palabras clave presentado en este trabajo. Se describió también en mayor profundidad el problema de la extracción de palabras clave y su relación con los problemas de clasificación presentes en la minería de datos convencional.

Se dio un marco de referencia al desarrollo de LIKE al mencionar y describir las técnicas de extracción de palabras clave más influyentes en el área. Muchos de estos métodos utilizan aprendizaje supervisado y dan al texto una representación vectorial calculando una serie de características, como la frecuencia en el documento y en el conjunto de documentos, la posición de la primera ocurrencia, la aparición en el título, etc. Al tratarse de aprendizaje supervisado se requiere etiquetar los ejemplos a ser usados en el entrenamiento del modelo de clasificación. Para este fin se utilizan las palabras clave designadas por los autores de los documentos de entrenamiento. Se vio que los diferentes métodos utilizan distintos algoritmos de aprendizaje como son los árboles de decisión, las redes neuronales y los clasificadores probabilísticos como Naive Bayes. Como se mencionó en el capítulo 2, en el desarrollo de LIKE se optó por las redes neuronales por su capacidad de generalizar y abstraer características complejas.

También en este capítulo se detalló la primera fase de LIKE, que consiste en procesar los documentos con el fin de obtener una representación vectorial de los términos a la que se pueda aplicar el algoritmo de aprendizaje elegido. La representación elegida para los términos es la de n-gramas, a los cuales se asocia un conjunto de características descriptivas. Se describió la etapa de la obtención de los n-gramas a partir del texto, la cual es una etapa importante no sólo por el tiempo de cómputo que exige, sino porque es una oportunidad de filtrar elementos poco frecuentes y por lo tanto poco relevantes para el entrenamiento.

Se describió el conjunto de características utilizado por LIKE, algunas de las cuales aparecen en otros algoritmos mencionados anteriormente. El objetivo es contar con un conjunto de características que sea capaz que describir las propiedades lingüísticas y estructurales de los términos y su utilización dentro del texto, de forma de poder abstraer las características que distinguen los términos que son palabras clave de aquéllos que no lo son. En resumen, las características utilizadas por LIKE son TF, TF-IDF, posición de la primera ocurrencia, posición promedio dentro de la oración, aparición en el título,

aparición de las palabras individuales en el título, longitud normalizada de la oración, y frecuencia normalizada (Z-Score). Todas estas características excepto TF-IDF y Z-Score están normalizadas entre 0 y 1. Además, entre ellas sólo el TF-IDF y el Z-Score dependen del cuerpo entero de documentos para su cálculo.

Teniendo la representación vectorial de los términos se puede entrenar la red neuronal *backpropagation* que va a actuar como clasificador. Esta etapa de entrenamiento junto con los problemas encontrados en ella, las pruebas realizadas y los resultados obtenidos van a ser detallados en el capítulo siguiente.

Capítulo 4

El método LIKE. Entrenamiento y evaluación del modelo

En este capítulo se describe la segunda fase de LIKE, la fase en la que se construye el modelo usado para clasificar los términos de un documento nuevo e identificar así las palabras clave. Para llevar a cabo esta tarea LIKE utiliza una red neuronal *backpropagation*, la cual es entrenada usando como ejemplos los vectores de características obtenidos en la fase de transformación del texto descripta en el capítulo anterior.

Los ejemplos están etiquetados en base a si los términos a los que corresponden fueron elegidos o no como palabras clave por los autores de los documentos. Naturalmente las palabras clave son mucho menos numerosas que el resto de los términos, con lo que la cantidad de elementos que pertenecen a la clase de las palabras clave es mucho menor que la cantidad de elementos de la otra clase. Esto hace que sea necesario aplicar una técnica de balanceo de clases antes de comenzar el entrenamiento.

Este capítulo está organizado de la siguiente manera. En la Sección 4.1 se describe el problema del desbalance de clases y la forma en que fue resuelto en el desarrollo de LIKE. En la Sección 4.2 se detalla la fase de entrenamiento de la red neuronal *backpropagation*, mencionando sus parámetros de configuración y otras consideraciones relativas a su aplicación. En la Sección 4.3 se describe en mayor detalle el algoritmo KEA, desarrollado por Witten y Frank mencionado en el capítulo anterior. Este es un algoritmo de extracción de palabras clave que utiliza aprendizaje supervisado, lo que hace interesante su comparación con LIKE por la similitud entre ambos métodos. En la Sección 4.4 se detallan las medidas utilizadas en las pruebas realizadas para evaluar el rendimiento de LIKE, más precisamente la precisión, el *recall*, y el F_1-*measure*. En la Sección 4.5 se describe el escenario de prueba utilizado para medir el rendimiento de ambos métodos y se muestran los resultados de estos experimentos.

4.1. El problema del desbalance de clases

El desbalance de clases tiene lugar cuando los elementos pertenecientes a una clase son mucho más numerosos que los pertenecientes a otra. Este problema dificulta los procesos de aprendizaje automático ya que muchos de los métodos de clasificación asumen que todas las clases tienen la misma importancia. Es por esta razón que en la mayoría de los problemas se recomienda construir los modelos utilizando datos representativos y en suficiente número de cada una de las clases. Sin embargo, en determinados contextos el desbalance de clases es natural al problema, como lo es en el caso de detección de fraudes o predicción de tormentas, en los que hay una clase de interés que tiene muchos menos elementos que la clase que no es de interés para el problema.

Los algoritmos de clasificación, y en particular las redes neuronales, se ven muy afectados por el desbalance de clases ya que durante el entrenamiento o construcción del modelo adquieren un sesgo hacia las características de la clase mayoritaria. Esto se debe a que el objetivo del algoritmo de aprendizaje es minimizar el error global de clasificación, sin tener en cuenta los errores de clasificación relativos de cada una de las clases. Así, en el entrenamiento se ignora el hecho de que el costo de clasificar mal una instancia que pertenece a la clase mayoritaria es mucho menor que el costo de clasificar mal una instancia de la clase minoritaria, que frecuentemente es la clase de interés para el problema.

Hay varias aproximaciones para resolver el problema del desbalance de clases. Algunas de estas aproximaciones se basan en aplicar un criterio de selección a los datos antes de comenzar el entrenamiento, mientras que otras se basan en modificar los algoritmos de entrenamiento para tomar en cuenta la naturaleza desbalanceada de la distribución de clases [28, 10].

Dentro del primer grupo las aproximaciones más representativas son el *random undersampling (RUS)* y el *random oversampling (ROS)* [20]. *Random undersampling* consiste en tomar una muestra aleatoria del conjunto de elementos de la clase mayoritaria y usar esta muestra en el entrenamiento. La muestra tomada debe tener aproximadamente el mismo tamaño que el conjunto de elementos de la clase minoritaria. Por su parte, *random oversampling* consiste en replicar elementos de la clase minoritaria hasta alcanzar un número semejante al de los elementos de la clase mayoritaria. Ambos métodos son simples de implementar y fáciles de aplicar. Sin embargo, *random undersampling* tiene la desventaja de que se pierde información de la clase mayoritaria, ya que se dejan elementos afuera del entrenamiento, y *random oversampling* tiene la desventaja de introducir redundancia en el conjunto de entrenamiento que antes no estaba presente, además de que el entrenamiento se hace más lento al contar con más ejemplos.

Otros enfoques proponen modificar los algoritmos de entrenamiento para tratar con el desbalance de clases [12, 10]. En el caso de las redes multiperceptrón hay modificaciones del algoritmo *backpropagation* que usan coeficientes diferentes para elementos de clases diferentes con el fin de asignar distintos grados de importancia a cada una de las clases y controlar la forma en que se modifican los pesos de la red neuronal. Si bien esto puede mejorar la clasificación de los elementos de la clase minoritaria, es difícil estimar el valor óptimo de estos coeficientes de forma de evitar el sobreajuste y además también es difícil generalizar esta técnica a n clases, en donde se tendría que calcular una matriz de costos de error de clasificación para cada clase. Otros algoritmos que buscan adaptar el proceso de entrenamiento para ponderar la importancia de las clases seleccionan dinámicamente los ejemplos de entrenamiento de acuerdo a un cálculo de la probabilidad que tiene cada ejemplo de contribuir a reducir los errores de clasificación de cada clase [28].

En el caso de LIKE el desbalance de clases se evidencia en que los términos que pertenecen a la clase de las palabras clave (los términos que efectivamente fueron elegidos por los autores de los documentos) son muy pocos en relación al resto de los términos (todos los demás términos que no fueron elegidos). En el desarrollo de LIKE se afrontó este problema mediante la aplicación de un algoritmo de agrupamiento.

El enfoque adoptado es similar al propuesto por Zhang [44], en el que se forman k grupos a partir de los datos de entrada, donde k es una constante arbitraria. Los grupos se obtienen mediante la aplicación del algoritmo K-medias, descripto en la Sección 2.4. Sea $nMin$ la cantidad total de instancias de la clase minoritaria y $nMax$ la cantidad total de instancias de la clase mayoritaria, se calcula para cada grupo i formado la proporción p_i de instancias de la clase mayoritaria dentro del grupo. Esta proporción se calcula de la forma $p_i = nMax_i/nMax$, donde $nMax_i$ es la cantidad de instancias de la clase mayoritaria en el grupo i. Luego se toman $p_i \times nMin$ elementos de la clase mayoritaria del grupo i, de forma de seleccionar un total de $nMin$ instancias de la clase mayoritaria. Estas instancias seleccionadas se van a usar junto con todas las instancias de la clase minoritaria en el entrenamiento del algoritmo de clasificación correspondiente. Los autores proponen dos medios para seleccionar las instancias dentro de cada grupo: selección aleatoria y selección por cercanía al centro del grupo.

En LIKE en cambio se forman $nMin/r$ grupos a partir de la clase mayoritaria solamente, y se seleccionan aleatoriamente r elementos de cada grupo, de forma de seleccionar en total también $nMin$ elementos. Los grupos se obtienen usando K-medias, al igual que en el enfoque de Zhang. El valor r se comporta como un factor de dispersión de la muestra a seleccionar ya que un valor alto de r hace que se formen pocos grupos y que los elementos extraídos sean similares entre sí al provenir del mismo grupo, y un valor bajo de r tiene el efecto contrario al formar más grupos y permitir potencialmente mayor diversidad en

la muestra. En última instancia, la cantidad de ejemplos extraídos depende de la cantidad de ejemplos en la clase minoritaria y la similitud de los ejemplos extraídos depende de la varianza de los mismos en el conjunto original.

La diferencia principal con el enfoque de Zhang es que en LIKE el agrupamiento se realiza solamente sobre la clase mayoritaria, ya que el fin buscado es resumir la información de esta clase. Como se espera que la clase de los términos que no son palabras clave sea diverso, el uso de muchos grupos y la selección aleatoria dentro de cada grupo permite reflejar tal diversidad a la vez que se balancea la distribución de clases y se reduce la cantidad de ejemplos a utilizar en el entrenamiento.

Una vez que se aplica el proceso de balanceo de clases se puede comenzar con el entrenamiento de la red neuronal para obtener el modelo de clasificación.

4.2. Entrenamiento de la red neuronal

En esta sección se describen las características de la red neuronal *backpropagation* utilizada como son la topología y los parámetros de configuración empleados.

La red neuronal utilizada consta de tres capas: la capa de entrada, la capa de salida y una capa oculta. La capa de entrada simplemente refleja el ingreso de una instancia a la red. Los vectores numéricos utilizados para representar los términos constan de ocho elementos, cada uno de los cuales corresponde a un atributo de los descriptos en la sección 3.3.2. La arquitectura utilizada se muestra en la figura 4.1.

La capa oculta está formada siete neuronas, cada una con ocho entradas a las que llegan las conexiones de la capa de entrada. El número de neuronas de la capa oculta, como se comentó en la sección 2.3.2, se determina empíricamente pero en LIKE se mantuvo la convención de que la cantidad de neuronas en la capa oculta sea menor que la cantidad de neuronas de la capa de entrada. La función de activación utilizada por las neuronas de la capa oculta es la función sigmoide o logística *(logsig)*, cuyo rango está entre 0 y 1, y que se visualiza en la Figura 2.3 en la sección 2.3.1.

En la capa de salida hay una única neurona con siete entradas, una para cada neurona de la capa oculta. Esta neurona es la responsable de combinar las salidas de todas las neuronas ocultas y determinar a qué clase pertenece el vector ingresado a la red. La función de activación utilizada por la neurona de salida es la tangente hiperbólica *(tansig)*, cuyo rango está entre -1 y 1. La utilización de esta función de activación hace que sea necesario que se cambie el valor de la clase mayoritaria (“no es palabra clave”) de 0 a -1, ya que de lo contrario la comparación de la salida de la red con la clase esperada nunca daría un valor

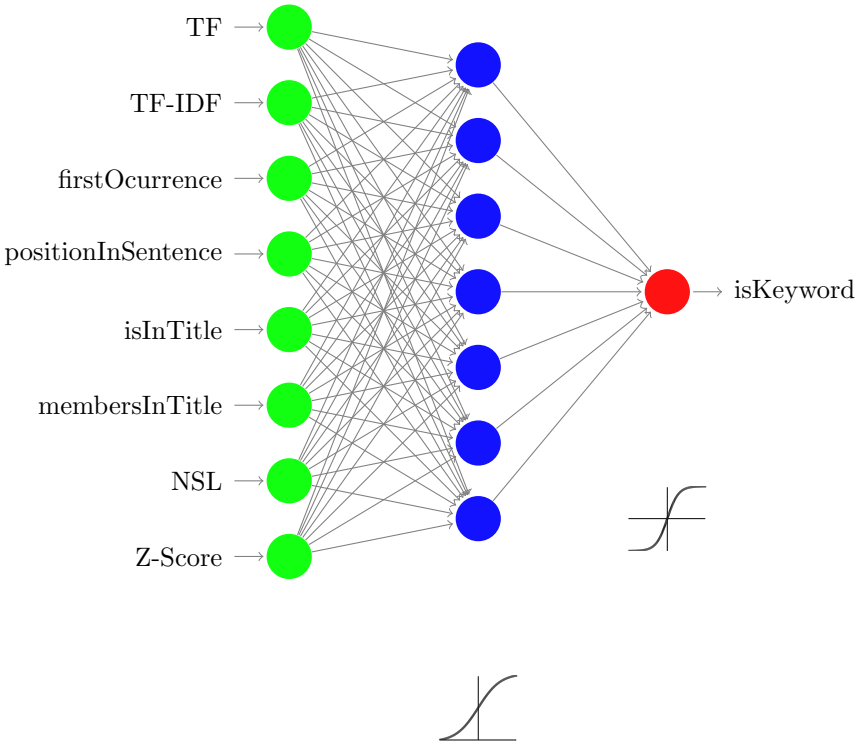

Figura 4.1: Red neuronal *backpropagation* utilizada como modelo de clasificación de palabras clave.

correcto. Esta función se ve en la Figura 2.4, también en la sección 2.3.1.

Se elige la tangente hiperbólica como función de activación porque tiene un rango más amplio y extremos más separados que la función sigmoide, con lo que se logra una mejor separación de las clases. Sin embargo, también se podría haber utilizado la función sigmoide en esta capa con lo que no sería necesario aplicar ninguna transformación a los valores de la clase mayoritaria. De igual forma se podría haber utilizado cualquier otra función de activación con la sustitución de los valores de salida que sea necesaria.

El factor de velocidad de aprendizaje utilizado es $\alpha = 0.25$. También se restringió el tiempo máximo de entrenamiento utilizando una cota de 1000 iteraciones, en cada una de las cuales se procesan todos los ejemplos de entrada. Alternativamente el entrenamiento finaliza cuando la diferencia entre la tasa de error obtenida en una iteración está por debajo de 1×10^{-8}. Esta tasa de error se calcula como el promedio entre los errores de clasificación de cada ejemplo (la diferencia entre el valor de clase esperado y la salida de la red) elevados al cuadrado.

Antes de comenzar el entrenamiento se escalan los ejemplos de forma de distribuir mejor

los valores de los atributos entre 0 y 1 y unificar la escala de los atributos TF-IDF y Z-Score que no están en este rango. De esta forma la combinación de los valores de entrada con los pesos de la neurona se encuentra en el rango de interés de la función de activación, que en el caso de la capa oculta es la función sigmoide.

El modelo de predicción obtenido es la red neuronal entrenada. Para aplicar el modelo a un documento nuevo se aplica el procesamiento de generación de características descripto en la Sección 3.3 y se ingresa cada vector obtenido a la red, la cual da como salida un número real. A la salida de la red se le aplica una función umbral que transforma el número real de salida en 1 o -1. Para las pruebas se utilizó el valor 0.6 como umbral, de forma que si la salida de la red es mayor o igual que este valor el algoritmo responde que el término dado es una palabra clave del documento y responde que no en caso contrario.

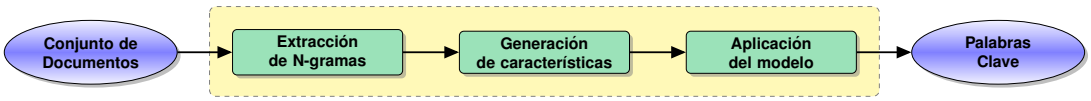

Figura 4.2: LIKE. Aplicación del modelo de clasificación

Los atributos TF-IDF y Z-Score son calculados utilizando el cuerpo de documentos de entrenamiento, lo que implica un costo extra en tiempo y en espacio ya que es necesario almacenar los documentos de entrenamiento en su estado original y volver a procesarlos para cada documento de prueba. Esta es la principal razón que motiva a considerar una segunda versión de LIKE, denominada LIKE-6 para distinguirla de la original (LIKE-8) y que no utiliza estos dos atributos. Para ver el efecto que tiene ignorar estos atributos en el rendimiento del método se realizaron pruebas con el fin de medir la precisión de LIKE-6 y la de LIKE-8.

4.3. KEA: Keyphrase Extraction Algorithm

En esta sección se describe KEA, el algoritmo de referencia usado en la evaluación del rendimiento de LIKE. KEA tiene características similares a LIKE como el hecho de que ambos son supervisados y requieren un conjunto de documentos de entrenamiento.

KEA *(Keyphrase Extraction Algorithm)* es un algoritmo de extracción automática de palabras clave que utiliza aprendizaje supervisado. KEA identifica secuencias de palabras candidatas a ser palabras clave, calcula para cada una de ellas una serie de características, y aplica Naive Bayes para predecir las candidatas que son buenas palabras clave [43].

La extracción es considerada como un problema de clasificación en el que cada término pertenece a una de dos clases: ser palabra clave o no. El algoritmo de aprendizaje utiliza como ejemplos los términos etiquetados con este valor de clase, identificando para ello aquellos términos que fueron designados palabras clave en los documentos de entrenamiento.

4.3.1. Generación de frases candidatas

Para realizar el entrenamiento KEA recibe como parámetros los documentos etiquetados con las palabras clave de los autores, un *stemmer* y una *stoplist*.

El primer paso de KEA consiste en dividir el texto de acuerdo a los delimitadores de oración (signos de puntuación, guiones, corchetes, apóstrofes y números). Los caracteres que no son alfanuméricos son removidos, a excepción de los puntos contenidos en palabras, como por ejemplo "C4.5".

KEA toma como frases candidatas las secuencias de palabras con las siguientes características:

- deben tener una longitud de a lo sumo tres palabras.
- no deben empezar o terminar con una *stopword*.
- no deben estar formadas por un único sustantivo propio, es decir, estar compuestos por una única palabra que aparece en mayúscula en todas sus ocurrencias.

En el siguiente paso se unifica la capitalización de las palabras y se aplica *stemming*, usando para el idioma inglés el algoritmo de Lovins [29] para eliminar sufijos. Luego de este paso se eliminan las candidatas que aparecen una sola vez en el texto.

4.3.2. Construcción del modelo

KEA utiliza como atributos de los términos a TF-IDF y la primera ocurrencia del término dentro del documento. TF-IDF mide la frecuencia del término dentro del documento en relación a su frecuencia en el cuerpo entero. Ambos atributos se calculan de la misma forma que en LIKE, y se describen en la sección 3.3.2.

Para aplicar Naive Bayes KEA discretiza los atributos usando el esquema de Fayyad e Irani [13] basado en el principio de longitud de descripción mínima *(Minimum Description Length)*. Este esquema divide recursivamente el atributo en intervalos,

minimizando en cada etapa la entropía de la distribución de clases. En otras palabras, el método busca que los intervalos sean lo más uniformes posibles respecto a la clase de los datos contenidos en cada uno de ellos.

Para aplicar Naive Bayes se asume que ambos atributos son independientes. Haciendo esta asunción dado un término con un valor discretizado T del atributo TD-IDF y un valor discretizado D del atributo de primera ocurrencia, la probabilidad de que un término sea palabra clave es:

$$P[key|T, D] = \frac{P[T|key] \times P[D|key] \times P[key]}{P[T, D]} \tag{4.1}$$

donde $P[T|key]$ es la probabilidad que una palabra clave tenga un valor T de TF-IDF, $P[D|key]$ es la probabilidad de la palabra clave tenga un valor D de primera ocurrencia, y $P[key]$ es la probabilidad *a priori* de que el término sea palabra clave. El factor $P[T, D]$ que surge del teorema de Bayes simplemente indica la probabilidad de obtener un término con las mismas características que el término observado, y puede ser omitido del cálculo ya que no es significativo para el cálculo de la probabilidad condicional. El resto de los factores de la fórmula se pueden estimar contando la frecuencia con la que aparece cada uno de los eventos.

De ésta forma la probabilidad de que un término sea palabra clave se puede calcular como:

$$P[key|T, D] = \frac{K}{K + N} P[T|key] \times P[D|key]$$

donde K es la cantidad de términos que son palabras clave en el conjunto de entrenamiento y N es la cantidad restante de términos que no lo son. La expresión $\frac{K}{K+N}$ es un estimador de la probabilidad de que un término cualquiera sea palabra clave, es decir, $P[key]$. La expresión $P[T|key]$ representa la proporción de los términos que son palabras clave y tienen un valor discretizado T en el atributo TF-IDF y lo mismo ocurre con $P[D|key]$.

De manera análoga se obtiene una expresión para calcular la probabilidad de que un término candidato no sea palabra clave:

$$P[no\_key|T, D] = \frac{N}{K + N} P[T|no\_key] \times P[D|no\_key]$$

La probabilidad final de que un término sea una palabra clave se calcula como

$$p = \frac{P[key|T, D]}{P[key|T, D] + P[no\_key|T, D]}$$

Este último es el valor utilizado como puntaje para *rankear* los términos, eligiendo aquéllos con la probabilidad más alta. Para armar el *ranking* se aplican dos pasos de posprocesamiento. En primer lugar, para desempatar dos términos con igual probabilidad se elige aquél que tenga mayor valor de TF-IDF sin discretizar. En segundo lugar se eliminan de la lista de términos aquéllos que estén contenidos en otros términos salvo que tengan un puntaje más alto que el término que los contiene. Por último, se eligen los r candidatos con el puntaje más alto, donde r es un parámetro del algoritmo suministrado por el usuario que representa la cantidad de palabras clave a extraer de cada documento.

Los autores de KEA proponen un tercer atributo que tiene el fin de aprovechar la información del dominio de los documentos de entrenamiento. Este atributo es la cantidad de veces que una palabra clave es elegida en el cuerpo de documentos *(keyphrase frequency)*, y refleja la noción de que si un término es elegido como palabra clave muchas veces es bastante probable que dicho término sea importante en el dominio al que pertenecen los documentos que lo tienen como palabra clave. Como este atributo es de valor entero se debe discretizar de la misma forma que los anteriores.

Entonces se puede reescribir la ecuación 4.1 de la siguiente manera:

$$P[key|T, D, F] = \frac{P[T|key] \times P[D|key] \times P[F|key] \times P[key]}{P[T, D, F]}$$

donde $P[F|key]$ es la probabilidad de que una palabra clave tenga un valor discretizado F como cantidad de veces que fue elegida. Naturalmente, este atributo sólo tiene sentido para documentos que pertenecen al mismo dominio, ya que de lo contrario la cantidad de veces que un término es elegido como palabra clave no tiene influencia en su elección en documentos de otros dominios.

Para aplicar el modelo de clasificación a un documento nuevo con el fin de extraer sus palabras claves KEA calcula el valor de TF-IDF, primera ocurrencia, y *keyphrase frequency* de todos los términos del documento nuevo. Luego el modelo determina para cada término la probabilidad de ser palabra clave y posteriormente la operación de posprocesamiento selecciona las mejores palabras claves del documento.

La versión 5.0 de KEA, lanzada en 2011, reemplaza el algoritmo de *stemming* de Lovins por el de Porter [37] para el idioma inglés, y agrega atributos adicionales de los términos como la longitud del término en palabras (el valor n del n-grama), la desviación estándar de la frecuencia del término, y la cantidad de términos que están relacionados semánticamente con el término dado. Este último atributo se calcula con la ayuda de un tesauro temático, el cual es un parámetro más del algoritmo [2].

4.4. Medidas de evaluación. Precisión, recall y F-measure

Para medir el rendimiento de los algoritmos de clasificación y también para las tareas de recuperación de información *(Information Retrieval)* se utilizan las medidas de precisión y *recall*. Ambas medidas evalúan la calidad de la clasificación de los elementos de una clase dada con respecto al resto de las clases.

Si se considera un problema de clasificación binaria en el que la clase de interés es la clase A, los resultados arrojados por un modelo de clasificación pueden pertenecer a una de cuatro categorías, como se ve en la matriz de confusión de la Tabla 4.1.

Real / Pred.	Clase A	Clase B
Clase A	TP	FP
Clase B	FN	TN

Tabla 4.1: Falsos positivos y falsos negativos

En la diagonal de la matriz de confusión están los aciertos del modelo, los verdaderos positivos *(True Positive, TP)* y los verdaderos negativos *(True Negative, TN)*. Los falsos positivos *(False Positive, FP)* ocurren cuando el modelo clasifica a un elemento como perteneciente a la clase de interés cuando no lo es, y los falsos negativos *(False Negative, FN)* ocurren cuando el modelo falla en clasificar correctamente los elementos de la clase interés. Así, los falsos positivos se pueden asociar a los errores de Tipo I en estadística (afirmar algo que es falso) y los falsos negativos se pueden asociar a los errores de Tipo II (negar algo que es cierto). Esto se puede extender a cualquier número de clases, considerando siempre una de ellas como la clase de interés.

Con estas definiciones se define la precisión como:

$$precision = \frac{TP}{TP + FP}$$

El *recall* se define como:

$$recall = \frac{TP}{TP + FN}$$

La precisión mide cuántos de los elementos identificados por el modelo son de interés mientras que el *recall* mide cuántos de los elementos de interés son identificados por el modelo. En cierta forma son inversamente proporcionales, ya que usualmente es fácil aumentar la precisión a expensas del *recall* y viceversa modificando la cantidad de elementos extraídos o identificados por el modelo.

Es común entonces que ambas medidas se combinen para dar una idea del rendimiento general del modelo. Para ello se utiliza la *F-measure*, que se define de la siguiente manera:

$$F_\beta = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 \times Precision + Recall}$$

La medida más utilizada es la F_1*-measure*, en la que $\beta = 1$ y por lo tanto precisión y *recall* tienen la misma importancia. La F_1*-measure* es equivalente a la media armónica entre la precisión y *recall*:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

4.5. Evaluación y resultados obtenidos

En esta sección se describen las pruebas realizadas y los resultados obtenidos sobre el rendimiento de LIKE. La implementación de KEA utilizada en las pruebas es la de los autores y se puede descargar libremente [3]. Es la implementación de la versión 5.0 e incluye *stoplists* y *stemmers* para varios idiomas.

Para las pruebas se utilizaron 96 artículos científicos de WICC en idioma inglés. Se aplicó *10-fold cross validation* como método de validación 30 veces de forma independiente sobre KEA y sobre LIKE. El procedimiento de *10-fold cross validation*, como se describe en la sección 1.2.4, consiste en dividir el conjunto de prueba en 10 partes iguales y usar 9 de ellas para entrenar el modelo y la parte restante para evaluarlo. Se realizan 10 iteraciones de este proceso seleccionando en cada una de ellas una parte diferente para la evaluación, y finalmente se promedian los 10 resultados obtenidos. Este procedimiento se efectuó 30 veces para obtener una muestra de 30 valores de precisión, *recall* y F_1*-measure* de LIKE y de KEA.

Los aciertos y los errores se calculan en base a la comparación de las palabras clave extraídas por ambos algoritmos y las palabras clave designadas por los autores. De esta forma una palabra clave extraída que no está entre las designadas por un autor es considerada un falso positivo, y una palabra clave de un autor que no está entre las extraídas por el modelo es considerada un falso negativo.

Se les aplicó un test de hipótesis de diferencia de medias a las muestras obtenidas para determinar si los resultados obtenidos en cuanto a las diferencias en precisión son estadísticamente significativos.

En la Tabla 4.2 se muestran los valores medios de precisión, *recall*, y F_1-measure

obtenidos promediando los valores de las muestras, y la desviación estándar de cada uno entre paréntesis [9]. La precisión media obtenida por LIKE, en sus variantes de 6 y 8 atributos, es superior a la obtenida por KEA y lo mismo ocurre con el *recall* y el F_1-*measure*. Estos valores se calcularon promediando los 30 valores de las muestras obtenidas en el paso anterior.

Se ve también que LIKE-6, la variante que no utiliza los atributos globales de TF-IDF y Z-Score, tiene menor precisión que LIKE-8 y un *recall* ligeramente menor. Esto da la pauta que estos atributos efectivamente contribuyen a la obtención de un mejor modelo de clasificación a expensas de un mayor costo computacional. Si la eficiencia del proceso de extracción es crucial LIKE-6 es una mejor opción, en cambio si se requiere obtener una mayor precisión es mejor utilizar LIKE-8.

	LIKE-8	LIKE-6	KEA
Precisión	0.76 (0.051)	0.65 (0.101)	0.52 (0.006)
Recall	0.75 (0.094)	0.72 (0.141)	0.37 (0.004)
F_1-*Measure*	0.74 (0.053)	0.68 (0.116)	0.43 (0.005)

Tabla 4.2: Valores medios de precisión, *recall* y F_1-measure para LIKE-8, LIKE-6 y KEA. La desviación estándar de los valores está indicada entre paréntesis.

En la Tabla 4.3 se ven los resultados de aplicar el test de hipótesis de diferencia de medias. Las muestras obtenidas anteriormente son comparadas en el test para determinar si estadísticamente las medias son diferentes, en cuyo caso se puede afirmar que un método tiene mayor precisión y *recall* que el otro.

	LIKE-8 vs. KEA	LIKE-6 vs. KEA	LIKE-8 vs. LIKE-6
Precisión	$(5{,}48 \times 10^{-22})$	$(5{,}19 \times 10^{-08})$	$(4{,}12 \times 10^{-06})$
Recall	$(1{,}50 \times 10^{-19})$	$(4{,}08 \times 10^{-14})$	(0,4832)
F_1-*Measure*	$(2{,}18 \times 10^{-24})$	$(2{,}51 \times 10^{-12})$	(0,0096)

Tabla 4.3: P-valores obtenidos al aplicar test de hipótesis con α=0.01

Los tests fueron realizados con un valor de significación de α=0.01. Si el p-valor obtenido es menor que el valor de α utilizado se puede rechazar la hipótesis nula (que las medias son iguales) en favor de la hipótesis alternativa (que las medias no son iguales). En casi todos los casos se observa que el p-valor está muy por debajo del valor de α con lo que se puede afirmar que tanto LIKE-8 como LIKE-6 presentan valores de precisión, *recall* y F_1-*measure* significativamente distintos que los obtenidos por KEA.

En cuanto a LIKE-8 vs. LIKE-6, se puede afirmar que la precisión de LIKE-8 es mayor que la de LIKE-6 y la evidencia es apenas suficiente para afirmar que el F_1-*measure* de LIKE-8 es mayor que el de LIKE-6. Por otro lado, no se puede afirmar que el *recall* de LIKE-8 sea diferente que el de LIKE-6, lo que confirma los resultados de la Tabla 4.2.

También se realizaron pruebas con 100 artículos en español provenientes de la misma base de datos, obteniendo resultados similares. Esto evidencia que aún contando con un conjunto de entrenamiento reducido LIKE es capaz de obtener una buena precisión, lo que no ocurre con KEA.

4.6. Resumen del capítulo

En este capítulo se describió la segunda fase de LIKE, la aplicación de la red neuronal como algoritmo de clasificación para obtener un modelo que permita obtener las palabras clave de un documento. Se describió el problema del desbalance de clases y la necesidad de resolver este problema antes de comenzar el entrenamiento. El enfoque adoptado en el desarrollo de LIKE consiste en aplicar un algoritmo de agrupamiento con el fin de resumir la información de la clase mayoritaria, en este caso la clase de los términos que no son palabras clave, y así facilitar el aprendizaje de las características de los elementos de la clase minoritaria, que es la clase de interés para el problema.

Se describió también la arquitectura de la red neuronal empleada y los parámetros de configuración de la misma, como la velocidad de aprendizaje y las funciones de activación usadas en cada capa. La red neuronal cuenta con ocho neuronas en la capa de entrada, cada una correspondiente a los atributos generados en la primera etapa del método descripta en el capítulo anterior. La capa oculta consta de siete neuronas y utiliza como función de activación la función sigmoide. La capa de salida está formada por una única neurona que utiliza como función de activación la tangente hiperbólica.

Se analizó en mayor profundidad el funcionamiento de KEA, el algoritmo contra el que se compara el rendimiento de LIKE. KEA utiliza aprendizaje supervisado, al igual que LIKE, y requiere por lo tanto un conjunto de documentos de entrenamiento manualmente etiquetados con las palabras clave elegidas por los autores de los documentos. KEA utiliza un conjunto menor de atributos de los términos, algunos de los cuales también usados por LIKE como el TF-IDF y la primera ocurrencia, y aplica Naive Bayes como algoritmo de aprendizaje.

Se detallaron las medidas utilizadas para evaluar el rendimiento de ambos métodos: precisión, *recall*, y F_1-*measure*. La precisión es la proporción de elementos identificados por el modelo de clasificación como elementos de la clase de interés que efectivamente

lo son, y el *recall* es la proporción de elementos de interés que son correctamente identificados por el modelo. En el caso de LIKE y KEA, la precisión mide la proporción de cuántos términos extraídos como palabras clave fueron elegidos también por los autores, y el *recall* mide la proporción de cuántas palabras clave designadas por los autores son también extraídas por los modelos. Lo ideal es que ambos valores altos pero generalmente ocurre que al subir uno disminuye el otro. Es por esta razón que para estimar con un único valor el rendimiento del algoritmo de clasificación comúnmente se utiliza el F_1-*measure*, que es el promedio armónico entre precisión y *recall*.

En este capítulo se detallaron también las pruebas realizadas y los resultados obtenidos. Las pruebas fueron realizadas utilizando 96 artículos científicos de WICC escritos en inglés. Se aplicó el método de evaluación *10-fold cross validation* unas 30 veces de forma independiente con el fin de obtener muestras a las que se pueda aplicar un test estadístico. Los resultados obtenidos muestran que la precisión y el *recall* obtenidos por LIKE son superiores a los de KEA para este conjunto de prueba.

También se discutió la aplicación de una segunda variante de LIKE, denominada LIKE-6, que no utiliza los atributos globales de TF-IDF y Z-Score. Si bien la omisión de estos atributos permite que el entrenamiento y la aplicación del modelo se realicen más rápido, los resultados de las pruebas muestran que la precisión y el *recall* alcanzados se ven afectados negativamente. Esto da la pauta que la elección de una u otra variante del método es una solución de compromiso entre la eficiencia y la precisión deseada.

Capítulo 5

Conclusiones

En esta tesina de grado se presentó un método de extracción automática de palabras clave a partir de documentos de texto. Esta es un área de investigación en auge debido a la gran cantidad de información textual disponible en la actualidad y la necesidad de facilitar los procesos de búsqueda y recuperación de este tipo de información.

La primera meta en el desarrollo de este trabajo fue definir una representación sobre la que se pueda aplicar una técnica de aprendizaje automático. Esta representación debe ser capaz de reflejar las características relevantes de los términos del texto.

En este trabajo se definió una representación que utiliza un conjunto de características relacionadas con la posición y la frecuencia de los términos. Usando esta representación se aplica una red neuronal a la que se proveen ejemplos de términos que son palabras clave y términos que no lo son. El resultado de este proceso es un modelo de clasificación capaz de determinar si un término dado es o no una palabra clave en base a sus características.

Uno de los mayores problemas que se presentaron en este trabajo fue el desbalance de clases, debido al reducido número de palabras clave en relación a la totalidad de los términos. Este problema fue resuelto aplicando un algoritmo de agrupamiento sobre la clase mayoritaria.

El método desarrollado fue aplicado a una base de datos real con resultados satisfactorios, habiendo alcanzado un rendimiento comparable al de uno de los métodos más citados en la literatura. Una de las principales ventajas del método propuesto es la independencia del idioma de los textos.

Los resultados de esta tarea de investigación han sido publicados y expuestos en el XIX Congreso Argentino de Ciencias de la Computación.

En el área de la extracción de palabras clave y de la minería de textos en general hay

todavía mucho por hacer. Como trabajo futuro se propone enriquecer la representación con otros atributos y analizar la importancia que tiene cada uno de ellos en la clasificación de los términos. También resulta de interés utilizar otros algoritmos de aprendizaje automático más complejos con el fin de obtener modelos más potentes.

Por otro lado, a futuro se tiene la intención de estudiar representaciones semánticas del texto más avanzadas que permitan mejorar la precisión de la extracción de palabras clave y realizar un análisis más profundo del sentido del texto.

Esto último constituye un cambio radical en el enfoque adoptado en esta tesina ya que el énfasis estaría puesto en la comprensión del texto más que en su construcción sintáctica.

Indice de figuras

Indice de tablas

Bibliografía

[1] ACM Computing Classification System. http://www.acm.org/about/class, 2013. Accedido en Agosto de 2013.

[2] KEA: Keyphrase Extraction Algorithm. http://www.nzdl.org/Kea/description.html, 2013. Accedido en Septiembre de 2013.

[3] KEA: Keyphrase Extraction Algorithm. https://code.google.com/p/kea-algorithm/downloads/list, 2013. Accedido en Septiembre de 2013.

[4] Proyecto de Enlace de Bibliotecas de la UNLP. http://prebi.unlp.edu.ar, 2013. Accedido en Agosto de 2013.

[5] Servicio de Difusión de la Creación Intelectual de la UNLP. http://sedici.unlp.edu.ar, 2013. Accedido en Agosto de 2013.

[6] Workshop de Investigadores en Ciencias de la Computación. http://redunci.info.unlp.edu.ar/wicc.html, 2013. Accedido en Agosto de 2013.

[7] Charu C. Aggarwal and ChengXiang Zhai, editors. *Mining Text Data*. Springer, 2012.

[8] Miguel A. Andrade and Alfonso Valencia. Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families. *Bioinformatics*, 14(7):600–607, 1998.

[9] Germán Aquino, Waldo Hasperué, César Estrebou, and Laura Lanzarini. A Novel, Language-Independent Keyword Extraction Method. In *XIX Congreso Argentino de Ciencias de la Computación (CACIC 2013)*, 2013.

[10] C.L. Castro and A.P. Braga. Novel Cost-Sensitive Approach to Improve the Multilayer Perceptron Performance on Imbalanced Data. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(6):888–899, 2013.

[11] Emilio Corchado and Colin Fyfe. *Introducción a la Minería de Datos*, chapter Redes Neuronales Artificiales. Editorial Pearson, 2004.

[12] Charles Elkan. The Foundations of Cost-Sensitive Learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[13] Usama M. Fayyad and Keki B. Irani. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *IJCAI*, pages 1022–1029, 1993.

[14] Ronen Feldman and James Sanger, editors. *The Text Mining Handbook. Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2007.

[15] Eibe Frank, Ian H. Witten, Gordon W. Paynter, Carl Gutwin, and Craig G. Nevill-Manning. Domain-Specific Keyphrase Extraction, 1999.

[16] James A. Freeman and David M. Skapura. *Neural Networks, Algorithms, Applications and Programming Techniques*. Editorial Addison-Wesley, 1991.

[17] Johannes Fürnkranz. A Study Using n-gram Features for Text Categorization, 1998.

[18] Stephen Grossberg. *Studies of Mind and Brain*. D. Reidel Publishing Company, 1982.

[19] Yaakov HaCohen-Kerner, Zuriel Gross, and Asaf Masa. Automatic extraction and learning of keyphrases from scientific articles. In *Proceedings of the 6th international conference on Computational Linguistics and Intelligent Text Processing*, CICLing'05, pages 657–669, Berlin, Heidelberg, 2005. Springer-Verlag.

[20] Haibo He and Edwardo A. Garcia. Learning from Imbalanced Data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009.

[21] Donald O. Hebb. *The Organisation of Behaviour*. Wiley, 1949.

[22] Robert Hecht-Nielsen. Counter-propagation networks. *Applied Optics*, 1987.

[23] José Hernández Orallo, M.José Ramírez Quintana, and César Ferri Ramírez. *Introducción a la Minería de Datos*. Editorial Pearson, 2004.

[24] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proc. of the 2003 Conf. on Empirical Methods in NLP*, pages 216–223, 2003.

[25] Pedro Isasi Viñuela and Inés M. Galván León. *Redes de neuronas artificiales: un enfoque práctico*. Pearson Educación - Prentice Hall, 2004.

[26] Jasmeen Kaur and Vishal Gupta. Effective Approaches for Extraction of Keywords. *IJCSI International Journal of Computer Science Issues*, 11 2010.

[27] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 1982.

[28] Minlong Lin, Ke Tang, and Xin Yao. Dynamic Sampling Approach to Training Neural Networks for Multiclass Imbalance Classification. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(4):647–660, 2013.

[29] Julie Beth Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[30] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297, 1967.

[31] Y. Matsuo and M. Ishizuka. Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information. *International Journal on Artificial Intelligence Tools*, 13, 2004.

[32] James L. McClelland, David E. Rumelhart, and The PDP Research Group. *Parallel Distributed Processing*, volume 1 and 2. MIT Press, 1986.

[33] R. Mihalcea and P. Tarau. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[34] Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, 1969.

[35] Dunja Mladenic and Marko Grobelnik. Word Sequences as Features in Text-Learning. In *In Proceedings of the 17th Electrotechnical and Computer Science Conference (ERK98*, pages 145–148, 1998.

[36] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web, 1999.

[37] M. F. Porter. Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[38] Frank Rosenblatt. The Perceptron – a perceiving and recognizing automaton, 1957.

[39] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. In *INFORMATION PROCESSING AND MANAGEMENT*, pages 513–523, 1988.

[40] Chade-Meng Tan, Yuan-Fang Wang, and Chan-Do Lee. The use of bigrams to enhance text categorization. *Inf. Process. Manage.*, 38(4):529–546, July 2002.

[41] Peter D. Turney. Learning Algorithms for Keyphrase Extraction. *INFORMATION RETRIEVAL*, 2:303–336, 2000.

[42] Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *IRE WESCON. Convention Record.*, volume 65, pages 96–104, 1960.

[43] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. KEA: Practical automatic keyphrase extraction. In *IN PROCEEDINGS OF THE 4TH ACM CONFERENCE ON DIGITAL LIBRARIES*, pages 254–255, 1998.

[44] Yan-ping Zhang, Li-Na Zhang, and Yong-Cheng Wang. Cluster-based majority under-sampling approaches for class imbalance learning. In *Information and Financial Engineering (ICIFE), 2010 2nd IEEE International Conference on*, pages 400–404, 2010.